

AFRL-ML-WP-TR-2001-4116

MEREOS



Charles W. Dement, Charles E. Mairet, Stephen E. DeWitt,
Robert W. Slusser

**ONTEK CORPORATION
540 POWER SPRINGS ST. SE
SUITE D23
MARIETTA, GA 30064-3562**

JUNE 2001

FINAL REPORT FOR PERIOD 09 JUNE 1995 - 18 JULY 2000

Approved for public release; distribution is unlimited.

**MATERIALS AND MANUFACTURING DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7750**


REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 01-06-2001		2. REPORT TYPE Final		3. DATES COVERED (FROM - TO) 09-06-1995 to 18-07-2000	
4. TITLE AND SUBTITLE MEREOS Unclassified			5a. CONTRACT NUMBER F33615-95-C-5519		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Dement, Charles W. ; Mairet, Charles E. ; DeWitt, Stephen E. ; Slusser, Robert W. ;			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME AND ADDRESS Ontek Corporation 540 Power Springs St. SE Suite D23 Marietta, GA30064-3562			8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS Materials and Manufacturing Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson AFB, OH45433-7750			10. SPONSOR/MONITOR'S ACRONYM(S)		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT APUBLIC RELEASE					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The MEREOS program is a manufacturing technology program of the United States Air Force Research Laboratory, Materials and Manufacturing Directorate, Manufacturing Technology Division (AFRL/MLM) funded under contract F33615-95-C-5519. The two program objectives were first, to identify the root causes of the problem of multiple BOM reconciliation and define the requirements for a solution, and second, to prove, via demonstration, the feasibility of developing a system to implement that solution. The key information system and enterprise process capabilities required to solve this and other related product representation problems were identified as a result of an extensive and formal needs and requirements analysis. Software demonstrating solution feasibility was also developed. Elements of a new aerospace and defense operating system for implementing these enterprise process capabilities were designed.					
15. SUBJECT TERMS product definition data management; product structure; multiple BOM problem; knowledge representation; enterprise engineering; enterprise process; enterprise operating system					
16. SECURITY CLASSIFICATION OF:		17. LIMITATION OF ABSTRACT		18. NUMBER OF PAGES	
		Same as Report (SAR)		109	
19. NAME OF RESPONSIBLE PERSON		Fenster, Lynn lfenster@dtic.mil			
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified	19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007		
					Standard Form 298 (Rev. 8-98) Prescribed by ANSI Std Z39.18

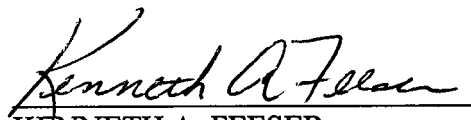
NOTICE

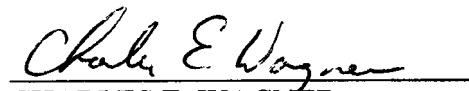
Using government drawings, specifications, or other data included in this document for any purpose other than government procurement does not in any way obligate the U.S. Government. The fact that the government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey and rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report has been reviewed by the Office of Public Affairs (ASC/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


JON F. JEFFRIES
Project Engineer
Industrial Infrastructure Team
Advanced Manufacturing Enterprise Branch


KENNETH A. FEESER
Team Leader
Industrial Infrastructure Team
Advanced Manufacturing Enterprise Branch


CHARLES E. WAGNER
Acting Chief
Advanced Manufacturing Enterprise Branch
Manufacturing Technology Division

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2001		3. REPORT TYPE AND DATES COVERED Final 06/09/1995 - 07/18/2000
4. TITLE AND SUBTITLE MEREOS			5. FUNDING NUMBERS C F33615-95-C-5519 PE 78011F PR 3095 TA 04 WU 42	
6. AUTHOR(S) Charles W. Dement, Charles E. Mairet, Stephen E. DeWitt, Robert W. Slusser				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Ontek Corporation 540 Power Springs St. SE Suite D23 Marietta, GA 30064-3562			8. PERFORMING ORGANIZATION REPORT NUMBER ONTEK/TR-01/002	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Materials and Manufacturing Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson Air Force Base, OH 45433-7750 POC: Jon F. Jeffries, AFRL/MLMS, 937-904-4353			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-ML-WP-TR-2001-4116	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 Words) The MEREOS program is a manufacturing technology program of the United States Air Force Research Laboratory, Materials and Manufacturing Directorate, Manufacturing Technology Division (AFRL/MLM) funded under contract F33615-95-C-5519. The two program objectives were first, to identify the root causes of the problem of multiple BOM reconciliation and define the requirements for a solution, and second, to prove, via demonstration, the feasibility of developing a system to implement that solution. The key information system and enterprise process capabilities required to solve this and other related product representation problems were identified as a result of an extensive and formal needs and requirements analysis. Software demonstrating solution feasibility was also developed. Elements of a new aerospace and defense operating system for implementing these enterprise process capabilities were designed.				
14. SUBJECT TERMS, product definition data management, product structure, multiple BOM problem, knowledge representation, enterprise engineering, enterprise process, enterprise operating system			15. NUMBER OF PAGES 112	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

CONTENTS

1	Summary	1
2	Introduction	3
2.1	PACIS Program Objectives	3
2.2	PACIS System Architecture	4
2.3	PACIS Project History	5
2.4	The Strategic Problem: Demonstrating PACIS	6
2.5	Selection Process	7
2.6	Program Execution and Evolution	9
2.6.1	Change 1: An Increase in Representational Amplitude	9
2.6.2	Change 2: A Shift in Technical Focus	9
2.6.3	Change 3: MEREOS Phase III, Redefined	12
3	Results Part I: Product Structure Definition and Management	15
3.1	Product Representation Needs	16
3.1.1	Structure Multiplicity	19
3.1.1.1	Representative Structure Types	19
3.1.1.2	Representative Structure Divergences	21
3.1.2	Amplitude	23
3.2	Product Representation Requirements	26
3.2.1	Metastructure	26
3.2.1.1	SOA Syntax and Notational Conventions	28
3.2.1.2	Schematic Metastructure Requirements	34
3.2.2	Core Metastructures	37
3.2.2.1	COMPOSITION	38
3.2.2.2	CONSTITUTION	39
3.2.2.3	INHERENCE	40
3.2.2.4	QUALIFICATION	42
3.2.2.5	QUANTIFICATION	43
3.2.3	Intrinsic Adjunct Metastructures	44
3.2.3.1	EQUIVALENCE	45
3.2.3.2	ALTERNATION	45
3.2.3.3	VARIATION	46
3.2.3.4	ORDER	46
3.2.3.5	TRANSFORMATION	48
3.2.4	Extrinsic Adjunct Metastructures	49
3.2.4.1	ARTICULATION	49
3.2.4.2	FACTORIZATION	50
3.2.4.3	CONSOLIDATION	50
3.2.5	Intertypic Adjunct Metastructures	51
3.2.5.1	INVOLVEMENT	51
3.2.5.2	CAPACITATION	52
3.2.5.3	REPRESENTATION	52
3.2.5.4	DESIGNATION	53
3.2.6	Table of Metastructure Schemata	54
4	Results Part II: Enterprise Operating System Definition	55
4.1	Enterprise Operating System Needs	57
4.1.1	Greater Product Realization Amplitude	57
4.1.2	Process Multi-Modality	58
4.1.3	Action Multilaterality	58
4.1.4	Totality and Autonomism	59

4.2	Enterprise Operating System Requirements	60
4.2.1	Methodological Requirements	60
4.2.2	Metastructure	63
4.2.3	Mission Definition	64
4.2.4	Enterprise Processes	67
4.2.4.1	Enterprise Process Characterizations	67
4.2.4.2	Intent Formalization	72
4.2.4.3	Effectiveness Measure Systematics	79
4.2.4.4	Operational Analysis and Strategic Contexts	80
5	Conclusions	88
6	References	90
7	Attachments	93
	Attachment 1:Comments Concerning STEP	94
	Attachment 2:A2 Format Technical Drawings	97

FIGURES

Figure 1:	Informatic Scale of a Large Multinational Enterprise	4
Figure 2:	Product Realization Process Elements	17
Figure 3:	Metastructures in Graphical Form	32
Figure 4:	Core Relations	37
Figure 5:	Equivalence Relations	45
Figure 6:	Alternation Relations	45
Figure 7:	Variation Relations	46
Figure 8:	Order Relations	46
Figure 9:	Evolution Relations	48
Figure 10:	Extrinsic Relations	49
Figure 11:	Articulation Relations	49
Figure 12:	Factorization Relations	50
Figure 13:	Consolidation Relations	50
Figure 14:	Involvement Relations	51
Figure 15:	Capacitation Relations	52
Figure 16:	Representation Relations	53
Figure 17:	Designation Relations	53
Figure 18:	Mission Definition Elements	65
Figure 19:	Principal Enterprise Processes	70
Figure 20:	The Capability Relation and Attributes	72
Figure 21:	Intent Formalization Elements	73
Figure 22:	Strategic Context Elements	81
Figure 23:	Effectivity Coordinates	83
Figure 24:	Location Coordinates	83
Figure 25:	Element Type Quadrants	84
Figure 26:	Strategic Contexts and Enterprise Process Definition Phases	85
Figure 27:	Contrast Variants	87
Figure 28:	CEA Variants	87
Figure 29:	Taxon Phylogeny and Taxonomy	98
Figure 30:	Architecture Element Elements	99
Figure 31:	Antecedence and Consequence Taxonomy	100
Figure 32:	Similarity Phylogeny and Taxonomy	101
Figure 33:	Metastructure Elements and Examples	102

TABLES

Table 1:	Product Representation Metastructure Relation Types	27
Table 2:	Core Relations	37
Table 3:	Intrinsic Relations	44
Table 4:	Extrinsic Relations	49
Table 5:	Intertypic Relations	51
Table 6:	Metastructure Schemata	54
Table 7:	Enterprise Operating System Definition Metastructures	63

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of several people whose contributions made the success of the MEREOS program possible. Lockheed Martin Aeronautical Systems in Marietta, GA provided significant financial support and technical resources, and many LMAS people were key supporters of this effort. Among these were Bill Kessler, C.T. “Tom” Burbage, Jon Hilton, Carol Donaldson, Don Meadows, Cheryl Bullard, and Bunny Clay; the latter two without whom nothing would have worked. We would especially like to thank Jim Hulsey and Ray Massey for teaching us the intricacies of the C-130 product definition and the mysteries of mod index drawings and the B18 baseline.

John Stanley, Mike Lee, and Diana Eppstein of Ontek have our thanks (and sympathy) for having developed PACIS and MEREOS software; there are few programmers in the world capable of rendering the extremely high-level abstractions involved into operational code, and these are three of the best. Mike Charnow lent his considerable expertise to these efforts and provided essential technical support. The “Ontek East” crew at Satyam Computer Services Ltd. in India, led by Kameshwara Kakaraparthi (alias “KRR”), also rendered invaluable development support. Small companies that are DoD prime contractors present finance people with unique challenges, which Nancy Loughrey supported by Dale Dement consistently met. Three honorary Ontekers and former Northrop employees who made critical contributions were Tom Griskey, who taught us Systems Engineering, and Heide Santos and Chris Conley, who 14 years ago taught us the intricacies of real BOMs, model effectiveness, and CPROS.

Alfred North Whitehead once wryly quipped that “It takes a unique mind to undertake an analysis of the obvious,” and Dave Smith of UCI and Peter Simons of Leeds University, both of whom have also been at Ontek for many years, are paradigms of such uniqueness. There is of course nothing obvious at all about the formal structures of Intentionality, the relations of Part and Whole, and Systematics. The value to the MEREOS program of Dave’s seminal work in phenomenology and Peter’s equally influential work in ontology and mereology, together with their 12 years of collaboration as the “Philosophy Division” of Ontek, cannot be overstated. Nor can that of Whitehead himself or that of Ernst Mayr, upon whose shoulders our formal system also stands, and to whom we owe a great intellectual debt.

Finally, we have been privileged to be Air Force ManTech contractors for 16 years, and during that time have enjoyed very fruitful and congenial relationships with many people in that organization. Notable among these are Gerry Shumaker, Dave Judson, Bill Schulz, Brian Stucke, Mickey Hitchcock, Steve LeClair, and especially Wally Patterson, who was both a tremendous advocate and exacting manager during his tenure Air Force project engineer for the MEREOS program. He has remained a trusted and respected friend ever since.

SUMMARY

The MEREOS¹ program is an Enabling Technology Developments and Demonstrations (ETDD) project of the Agile Manufacturing Pilot Program (AMPP) funded by the United States Air Force Research Laboratory, Materials & Manufacturing Directorate, Manufacturing Technology Division (AFRL/MLM) under contract F33615-95-C-5519.

MEREOS is one element of the Platform for Automated Construction of Intelligent Systems (PACIS[®]) project. The two primary MEREOS program objectives were first, to identify the root causes of the multiple BOM reconciliation problem and define the requirements for a solution, and second, to prove, via demonstration, the feasibility of developing a system to implement that solution.

The technical aim of the PACIS project is to produce a next-generation database programming system that provides the tools required to develop, deploy and sustain large-scale manufacturing enterprise decision automation systems. The strategic aim of the project is to enhance the value of solutions enterprises deliver to their stakeholders by bringing automation to bear on enterprise integration and on critical processes in their value streams—specifically, those processes requiring extensive experience and cognitive skills to execute or govern—and, accordingly, those which have heretofore been intractable to any but the most superficial automation. Launched in 1981 as a privately funded effort led by the team who founded Ontek Corporation in 1985,² the PACIS project has been funded since then by AFRL/MLM and several industrial organizations.³

¹ An ungrammatical plural rendering of the Greek term μέρος (*part*)—thus, “parts.”

² The project originated as an effort to develop a decision automation system for Reynolds & Taylor, Inc., a second tier aerospace & defense subcontractor located in Santa Ana, California.

³ Alcoa, Northrop Aircraft Division, Westinghouse Electronic Systems Group, and Lockheed Martin Aeronautical Systems. Beginning in late 1999, the Lockheed aeronautics sector companies were consolidated into a single company called LM Aero, which is headquartered in Ft. Worth, Texas.

Information is the raw material of decision-making, and information systems are instrumentalities for provisioning decision-making processes—they are tools for provisioning purposeful thought. The value of any particular tool to a potential user ultimately consists in two attributes. The first is its applicability to the aims of a user—its *utility*. The second is its capability to facilitate accomplishment of those aims—its *effectiveness*. In the final analysis, the value of a tool can only be gauged by using it.

PACIS is a tool for creating decision automation systems—specific types of information systems—it is a tool *for toolmakers*. Its value accordingly consists in its instrumental applicability to the aims of *those* users and in its effectiveness in enabling accomplishment of *their* aims. This makes it difficult if not impossible to demonstrate its value to executive management and other indirect beneficiaries of the technology. To overcome this inherent ‘value visibility’ problem, we have over the years produced and demonstrated solutions to *visible* albeit technical needs of those who would ultimately benefit from decision automation systems: executive management and support staff. MEREOS is the most recent of these.

MEREOS program execution was structured into three distinct phases in accordance with ETDD/AMPP project requirements. These were:

Phase I	Technology Feasibility	Dec 94–Dec 95
Phase II	Prototype Demonstration	Jan 96–Dec 98
Phase III	Pilot Production Implementation	Jan 99–Jan 00

The original MEREOS Statement of Work (sow) addressed Phase I and II tasking. Phase III was an option which required separate authorization and a second specific sow. All three phases were executed.

The MEREOS program was originally conceived as an information systems-centric effort to identify the requirements for and demonstrate a solution to the multiple BOM reconciliation problem. While these original program objectives were achieved, the program underwent a transformation in both focus and venue during its execution. We set out to solve an informatics problem: we accomplished that, but also ended up solving several enterprise architecture problems as well, resulting in a baseline for a new aerospace enterprise operating system. The reasons for this ostensibly radical shift are discussed at some length in paragraph 2.6 in the Introduction.

Principal program results consist in formal requirements definitions for two specific systems. One was for a new kind of product data management system. The other was for a new kind of enterprise operating system. The first was demonstrated by software. The second was demonstrated by execution of an enterprise architecture program. Each are directly related to the other, albeit in complex and intricate ways. The former results are complete, and are presented in Section 3: Results Part I. The latter is a work in progress, although by the end of the program a great deal of work had been accomplished, and a great many lessons had been learned. The results obtained as of December 2000 are presented in Section 4: Results Part II.

■ ■ ■ ■

“Intelligence is the faculty of making artificial objects, especially tools to make tools.”

Henri Bergson, *L'Evolution Creatrice*

2

INTRODUCTION

The MEREOS program is one element in a larger multi-year effort called the PACIS project. It was designed to serve three purposes in that context. The first was to demonstrate, via a specifically targeted application, the technical capabilities and strategic business value of PACIS technology to potential user organizations. The second was to provide a technical focus for a major re-design of PACIS representation system metastructures. The third was to provide a suitably complex test bed for internal evaluations of PACIS representation system capabilities and performance.

2.1 PACIS Program Objectives

The goal of the PACIS project is to produce a next-generation database programming environment that provides the tools required to develop, implement and sustain enterprise management decision automation systems. A decision automation system is a large, geographically distributed intelligent computing system designed to automate many of the management support staff functions currently performed by human beings. One can think of decision automation systems as white collar robots and PACIS as a tool for building and maintaining them.

The specific objective of a management decision automation system is to enhance the economic viability and agility of an enterprise by automating tasks that require extensive expertise, such as change impact analysis, resource allocation, tactical plan development, make/buy decisions, and so on. Existing enterprise systems do not automate these kinds of activities. At best they provide some of the more mundane information management and computation services that people need to perform them. The high non-touch labor cost content of total product cost (60 - 70%) in most manufacturing industry sectors is one symptom of this phenomenon; the tremendous costs and long lead times associated with incorporating engineering changes into complex products is another.

The central tenet underlying our approach to PACIS is that decision automation systems cannot be

built, let alone implemented and sustained, without extensive automation. That is, the tasks of constructing, implementing, and sustaining decision automation systems must themselves be automated. The two most daunting characteristics of management decision automation systems are their physical and conceptual scales. The existing automated information assets of a reasonable size enterprise are measured in the multiple-terabytes, as the figure below illustrates.⁴

#	Description Of Metric	Value
1.	Total Number Of Distinct Programs	307,350
2.	Total Number Of Lines Of Source Code	288,391,200
3.	Total Number Of Information Entity Databases	14,130
4.	Total Number Of Information Entities	20,790
5.	Total Number Of Information Entity Data Elements	401,670

Figure 1. Informatic Scale of a Large Multinational Enterprise

Any management decision automation system would have to manage a substantially larger information base. Thus, the physical scale is well beyond the current approach to database management. The conceptual scale, required to support management decision automation, is likewise beyond the current capabilities for representing the knowledge of an enterprise. A system providing a fusion of these capabilities at the required physical and conceptual scales does not, as yet, exist.

2.2 PACIS System Architecture

PACIS is made up of four component subsystems. These are:

1. Nucleus

The PACIS nucleus is a portable distributed virtual machine providing a comprehensive suite of computing services to the other PACIS subsystems. These services include host computing platform-independent program invocation and runtime management, memory management, I/O, and network communications.

2. Information management subsystem

The PACIS information management subsystem is an ANSI[1] and ISO[9] 3-schema compliant DBMS providing a unified information definition, delivery, and management environment supporting transparent and dynamic access to information contained in heterogeneous databases residing on distributed heterogeneous computing platforms. The core of this subsystem is a knowledge representation system that supports, among other things, representations of processes, and intensional contexts as first-class entities; direct representation and manipulation of foreign system schemata and operations; and a comprehensive classification system that supports the definition and manipulation of distinct kinds of taxonomic schemes in several alternative topologies.

3. Presentation subsystem

The PACIS presentation subsystem contains two distinct components: the human interface system and the application program interface. The human interface system provides an

⁴ These numbers represent estimates we developed during our participation in the Alcoa Information Architecture program from 1989–1991. They are of course outdated now; the current physical scale of Alcoa applications and databases would of course be *much* larger than this.

extensive suite of interactive graphical interfaces and hardcopy reporting facilities required for user interaction with PACIS. The application programming interface is a library of callable functions and routines that enable third-party system developers to access the PACIS nucleus and information management subsystem services from their applications.

4. Semantic repository

The PACIS semantic repository is a collection of knowledge bases containing definitions for a broad class of entities commonly encountered in business environments.

The current version of PACIS is based upon over 15 years of research and development whose goal is the fusion of knowledge representation, database management systems, and declarative programming technologies.

2.3 PACIS Project History

The PACIS project began in 1981 as an effort to develop a decision automation system for Reynolds & Taylor, a second-tier aerospace job shop employing 250 people, located in Santa Anna, California. By 1985, a version of that system had been developed and successfully implemented in the company. The Reynolds & Taylor system was a fully integrated job shop management system that provided extensive automation for tasks such as estimating, process planning, material management, MIL-Q-9858 compliant quality management, and FAR/DFARS compliant finance and administration. This system was in turn built using a precursor of PACIS developed between 1981 and 1984. The system was used to operate Reynolds & Taylor until the company was sold in 1989.

Because the Reynolds & Taylor system provided functionality unavailable in other much larger systems,⁵ the project attracted the attention of both the Air Force and several prime contractors—notably Northrop Aircraft Division, Alcoa, and Westinghouse Electronic Systems Group—who were customers of Reynolds & Taylor. As a result, in late 1985 the Air Force Materials Laboratory, Manufacturing Technology division (now AFRL/MLMS) awarded Reynolds & Taylor a Small Business Innovation Research (SBIR) contract to improve the capabilities of the original tools used to build the system and scale them to the point where they could be used to develop similar systems for prime contractors. In response to this award, matching funds were provided by Alcoa, Northrop, and Westinghouse, the original project team formed Ontek Corporation, and the SBIR contract was novated from Reynolds & Taylor to the new company.

From 1986 to 1988, the team conducted fundamental research in knowledge representation, focusing on developing techniques for encoding and manipulating the complex cognitive processes and semantic structures involved in manufacturing enterprise management. This work culminated in the development of a prototype representation system which was demonstrated to a large industry and DoD group in early 1988. Based upon that demonstration, Northrop awarded Ontek a major subcontract under the Automated Airframe Assembly Program (AAAP) to develop a full-scale version of the system—by then designated PACIS. The first working version of PACIS was delivered to Northrop in December of 1989.

The first version of PACIS was used to develop several experimental application systems, each designed to test and demonstrate a key capability required for its eventual use as a platform for management automation systems. Three of these were:

- Concurrent engineering model integration system; an application for modelling and analysis of the To-Be configuration of the Northrop Product Definition and Development Center (PDDC), the Integrated Product Definition (IPD) organization for the F-23 program. This system was developed to demonstrate the ability of PACIS to represent complex processes and conceptual structures, and was used to model program organization structure, activity flow, documentation trees, and product structures in a single unified environment, and supported interactive graphical analysis and simulation of these structures.

⁵ Raw material lot and batch number traceability by part serial number; actual cost by part serial number; shop scheduling to the machine tool/operator level; non-destructively updated databases; fully dynamic user-definable ad-hoc queries; etc.

- Legacy database integration system; an application supporting dynamic ad-hoc queries across distributed heterogeneous databases, including IMS, DB2, and ORACLE. One of the major subsystems in this application was an IMS schema acquisition system, which was developed to demonstrate the ability of PACIS to subsume other database management system schemata. This subsystem automatically generated PACIS representations of IMS database systems directly from compiled binary on IBM mainframes. The legacy database access system was used to support several other demonstration applications at Northrop under the AAAP, at Alcoa, and at Westinghouse.
- Material status reporting system; an application developed for the Westinghouse Advanced Development Organization (ADO) to provide material status accounting on outside purchases and inventory for prototype product development. This system was developed to demonstrate the ability of PACIS to provide commercial third-party applications with transparent access to legacy databases. It enabled end users to directly access information contained in both IMS and DB2 databases via EXCEL spreadsheets on a Macintosh. Prior to the implementation of the application, this process took one person up to 5 days of elapsed time to perform. The application produced more accurate information than the manual procedure and required an average of 15 minutes of elapsed time to execute.

Work began in late 1991 to transition the project from a focus on R&D to full-scale development of a production version of the system. These activities were funded by AFRL/MLMS under the Corporate Data Integration Tools (CDIT) program, with additional funding from Alcoa and Westinghouse, and continuing technical support from Northrop. The objectives of these transitional activities were to develop a formal methodology for semantic analysis, to support the creation of PACIS knowledge bases for use in an enterprise integration environment, and to develop new PACIS interfaces to support the methodology. By the end of 1993, the new analysis methodology had been developed, a new suite of interface tools to support PACIS knowledge base definition had been produced, and preliminary high-level designs for a new production version of the system had been developed. Detailed design and preliminary development of the new production version of PACIS began in January of 1994, funded by WL/MT under the Advanced Enterprise Integration (AEIP) program.

The overall goals of the AEIP program were to improve PACIS system performance and maintainability, and to develop the software infrastructure required to implement a database programming language (DBPL). Three specific objectives were established to support these goals: redesign of the PACIS nucleus; design for the PACIS DBPL; and, selection of a suitable demonstration application for the new version of PACIS.

2.4 The Strategic Problem: Demonstrating PACIS

Tools, whether concrete or abstract, are instrumentalities for performing actions. Tools are a means to achieve certain ends, and are not, in that context, ends in and of themselves. The efficacy of a tool is a measure of two fundamental characteristics: the range of actions the tool enables its users to execute—its amplitude—and the efficiencies one obtains by using that tool—its effectiveness. But because a tool is a means to an end, the overall value of a tool is inherently indirect; that is, its value is dependent upon the value of the products that are produced by using it.

Information systems are tools that are used by people to operate an enterprise. Like any other tool, the value of an information system can only be determined indirectly; that is, by measuring the degree to which using that system improves the ability of people to achieve the goals of the enterprise they operate. However, information systems are inherently very complex; they are more like 5-axis spar mills than hammers. Because of this, information systems require a multitude of subsidiary tools in order for them to be economically built and maintained. Among these are such subsystems as operating systems, compilers, and, of course, database management systems.

Database management systems are instrumentalities for performing storage and retrieval operations on the data that the information systems they host acquire, manipulate, and present to their users. And, modern large-scale information systems simply could not be built, let alone maintained, without database management systems. So database management systems are tools. But because

database management systems are tools for building and maintaining other tools—namely information systems—their value is indirect, and is, accordingly, difficult to ascertain.

PACIS is a database programming system, hence it is a tool for building tools—information systems—and therefore presents the same difficulties in determining its value as does any other DBMS. However, PACIS differs from existing database management systems in several important ways. Collectively, these differences make the task of determining an unambiguous measure of its value even more difficult than it is for existing database management systems. Nevertheless, it is essential that a means of determining the value of PACIS to an enterprise be developed, and that its efficacy in that context be measured and documented.

The best way to demonstrate the value of a tool is to use it to actually perform the operation or operations it was intended to support, measure the quantitative features of the process one expects the tool to effect, and compare these to a known baseline, thereby determining the value of the tool. Consider: if one is going to determine the value of a 5-axis spar mill, one must, at some point, actually use the machine to make some wing spars. One must also, of course, make sure while doing so to measure the amount of time and quantity of resources the mill consumes to make the spars. Assuming one already had measured the time and resources required to perform the same task using another tool, one can then compare the two, and thereby determine a relative value.

The strategy adopted to measure and document the value of PACIS exploits the fact that PACIS—like any other DBMS—is, fundamentally, a tool for building and maintaining information systems. Thus plans were formulated, under the auspices of a successor program to the AEIP, to develop a product definition management application system—designated MEREOS—in order to demonstrate PACIS functionality and value.

The goal of MEREOS was to solve the multiple bill of materials (BOM) reconciliation problem in large-scale, technically advanced complex product manufacturing environments. The specific objectives were to provide end users with the ability to define, modify, query, and automatically maintain relationships between several distinct BOMs, specification trees, and functional architectures for a single product, where the information involved is stored in geographically distributed heterogeneous databases. The approach was to use PACIS to demonstrate a product data management system (PDM) meeting these objectives.

2.5 Selection Process

As stated earlier, the motivation for the MEREOS project was to demonstrate the capability and value of the PACIS technology and associated methodologies. The decision to develop a product definition management application system for this purpose, as opposed to any number of other applications one could develop with PACIS, was based on several interacting factors and desiderata.

The most important criterion determining the decision was that the demonstration application system provide a solution to a problem in the manufacturing industry. Specifically, the problem had to satisfy the following six criteria:

1. The problem had to be well known—i.e., the problem was familiar throughout the manufacturing industry.
2. The problem was endemic to both the defense and commercial sectors of the manufacturing industry.
3. In the case of the defense industry, the problem had to be visible to the customer with direct (and negative) impacts on customer life-cycle costs.
4. The problem had to be tractable to automation and Ontek personnel had to have the technical expertise and experience required to solve the problem.
5. Solutions to the problem in the form of software could not already exist.
6. There had to be direct, measurable, negative effects on product cost and schedule traceable to the problem, and the benefits to cost and schedule of solving the problem had to be equally direct, measurable and traceable.

The multiple BOM reconciliation problem stood out from the others by significant measure in all the above considerations. A viable technical solution to this problem has eluded all manufacturing companies, despite many information system-based attempts over four decades. Although there was an almost unanimous industry consensus concerning the importance of this problem, the root cause problem was not well understood, nor had requirements for a feasible solution been identified in any systematic way. It represented the kind of challenge needed to further PACIS technology and methodology developments, as well as to demonstrate their combined value.

Almost all manufacturing enterprises producing complex products also develop separate engineering (E-BOMS), manufacturing (M-BOMS), and logistical or field support (L-BOMS)—representing ‘as-designed’, ‘as-planned’, and ‘as-supported’ product configurations—in order to support various engineering, manufacturing, and maintenance activities. Further, each of these configurations inevitably differ from one another both in form and content. For example, in order to accommodate empirical producibility constraints, one part on a product E-BOM is sometimes made from two or more parts found only on the product M-BOM. In this case, the E-BOM will designate only one component, whereas the M-BOM will designate at least three. This kind of divergence marks the presence of a relation we call *articulation*. Conversely, it is sometimes possible to make several parts on an E-BOM from one single ‘progenitor’ part, thereby enhancing process or material utilization efficiency. In this case, the E-BOM for that product will only designate these particular components, but the M-BOM will also designate the progenitor *and* the relationships between it and the particular components. This kind of divergence marks the existence of a relation we call *factorization*. Finally, many parts on a product E-BOM and its M-BOM—frequently entire assemblies—never appear at all in its L-BOM, because once assembled, the components are not non-destructively accessible. This kind of divergence signifies the presence of a relation we call *integration*.

The existence of these divergences among BOMS together with the correlated but implicit relations between them is the genesis of the multiple BOM reconciliation problem. That is, the task of reconciling multiple BOMS for a product involves identifying components that stand in these “counterpart” relations across them, and characterizing the type and characteristics of those relations. Establishing counterpart traceability is, in turn, essential for executing and propagating the consequences of engineering change. Managing this process is possibly the most complex, costly, and error-prone activity in any manufacturing enterprise. Identifying and accommodating the ramifications of even a single modification to one component in one product BOM often requires the coordinated expertise of several speciality disciplines, such as materials and process, mechanical, electrical, and software engineering.⁶ The multiple BOM phenomenon exacerbates this already difficult problem, since the impacts of changes to a component in one BOM must be determined for all of its counterparts in any other BOMS. Thus, there is a direct linkage between the multiple BOM reconciliation problem and the high costs, long lead times, and error rates associated with engineering change.

Accordingly, this and other related problems have received a great deal of attention by the manufacturing and software industries, and by the DoD. There has been, and is now, a literal constellation of initiatives, software products, and standardization efforts designed to address product representation issues.⁷ Nevertheless, none so far have explicitly addressed the BOM reconciliation problem.

The differences that exist between these BOMS exist for very good business reasons. However, absent the ability to properly identify and characterize these differences by other than essentially manual procedures, they become significant cost, schedule, and quality drivers for engineering change in specific and all product operations and support activities in general. Moreover, the detrimental effects of this problem are familiar throughout both the defense and commercial sectors of

⁶ According to an article in *Fortune*[9] “Boeing’s fusty production techniques carry an enormous price tag. Every alteration, even a seemingly minor one like moving the location of an emergency flashlight holder, consumes thousands of hours of engineering time, requires hundreds of pages of detailed drawings, and costs hundreds of thousands, if not millions, of dollars to execute.” The flashlight example is perhaps a bit overstated, although the point is, in essence, valid. This situation is not unique to Boeing, nor is it unique to the aerospace and defense industry.

⁷ A notable example of these is PDES/STEP. We have included a brief outline of STEP as it pertains to specific issues addressed by the MEREOS program in *Attachment 1: Comments Concerning STEP*.

the manufacturing industry. Moreover, these effects were considered measurable (at least with some effort). We viewed the problem as tractable to solution using PACIS technology. Ontek personnel have had a great deal of prior experience with this problem domain. A solution to the problem required the full representational power of PACIS, and thus was considered a good demonstration of PACIS functionality. And, no system existed that was specifically designed to address the root cause of the problem. Finally, the multiple BOM reconciliation problem satisfied all 6 of the criteria listed above. Accordingly, it was chosen as the application domain to use to demonstrate the new version of PACIS.

2.6 Program Execution and Evolution

MEREOS program execution was structured into three distinct phases in accordance with Enabling Technology Developments and Demonstrations Agile Manufacturing Pilot Program (ETDD/AMPP) project requirements. These were:

Phase I	Technology Feasibility	Dec 94–Dec 95
Phase II	Prototype Demonstration	Jan 96–Dec 98
Phase III	Pilot Production Implementation	Jan 99–Jan 00

The original MEREOS Statement of Work (sow) addressed Phase I and II tasking. Phase III was an option which required separate authorization and a second specific sow. All three phases were executed.

The MEREOS program was originally conceived as an information systems-centric effort to identify the requirements for and demonstrate a solution to the multiple BOM reconciliation problem. While these original program objectives were accomplished, the program underwent several significant changes both in focus and application venue during its execution. We set out to solve a difficult and long-standing informatics problem—we accomplished that, but also ended up solving several enterprise systematics problems as well, creating a baseline for a new aerospace enterprise operating system. This shift warrants discussion.

2.6.1 Change 1: An Increase in Representational Amplitude

Preliminary analysis of the root causes of the multiple BOM reconciliation problem had been done prior to Phase I. This work was documented and subsequently published by Kluwer in 1995 as *Aspects of the Mereology of Artifacts* [20]. Phase I sow tasking provided for more comprehensive and detailed analysis of the problem, definition of requirements for a solution, and research and development activities to demonstrate solution feasibility.

The first change involved an increase in the representational gamut planned for the Phase II system, which occurred as a result of Phase I findings. One specific Phase I determination was that an effective solution to the multiple BOM reconciliation problem necessitated mechanisms for explicitly representing traceability of product structures to so-called ‘design intent’—that is, traceability of product definition elements, represented by various BOMs and document trees, to functional, producibility, and maintainability requirements. This in turn entailed the capability to explicitly represent relevant products of the systems engineering Needs Identification, Requirements Analysis, and Synthesis processes, namely technical requirements and their relations to product definition elements. This ‘upstream’ increase in representation amplitude was required to provision ‘downstream’ product structure definition management—specifically, engineering change impact identification and analysis.⁸

2.6.2 Change 2: A Shift in Technical Focus

The second change occurred in the early stages of Phase II, and represented a shift in technical conception and focus rather than in breadth. Its impetus was partly due to the increase in representa-

⁸ The need to explicitly represent technical requirements and their relationships to product structures is discussed in Section 3: Results Part I, paragraph 3.1.2.

tional amplitude mentioned above. But this shift was also driven by a need to address an increasingly exasperating strategic positioning problem, itself symptomatic of what we call the Commercial Off-The-Shelf (cots) problem.

MEREOS software was originally envisioned to be a stand-alone PDM application system for demonstrating multiple BOM definition and reconciliation capabilities to end users, thereby illustrating the value of PACIS technology to those stakeholders. However, the increase in representational scope required to completely solve that problem was significantly greater than originally anticipated, and accommodating this change in turn forced a change in the level of effort mix between formal systematics and programming. Constrained by fixed resources and schedule commitments, greater emphasis on formal systematics had to be compensated for by a reciprocal reduction in software development. Reducing software development level of effort without sacrificing the core technical capabilities required to achieve our solution demonstration objective necessitated a software architecture change—a change from an end user-oriented application to a developer-oriented tool, focused exclusively on providing application-independent product structure representation and integrity management services. We retained the effort to develop interfaces for visualizing these structures, and we of course retained the development effort required to implement the representation constructs required; this was, after all, the critical element of the engineering effort. However, we abandoned sweeping enhancements to the I/O and foreign system integration subsystems we had originally envisioned developing, along with development of a new interpreter and compiler for the PACIS database programming language, as neither of these would, in and of themselves, convey visible value to end user organizations. Although frustrating at the time, this necessary change in focus turned out to have been very fortuitous indeed, for both technical and strategic reasons.

One strategic benefit of the focus change and the resulting architectural repositioning was made abundantly clear during a meeting we held in early September 1996. The objectives of this meeting were to present to industry the results of our multiple BOM problem analysis, demonstrate a prototype illustrating key elements of a solution, and solicit a demonstration site for Phase III of the program. The meeting was attended by 18 organizations including all the major aerospace and defense prime contractors, Deere, and Caterpillar. The results of this meeting were simply astonishing. They were also ominous from a business point of view. The upshot was this: no one seriously challenged the results of our problem analysis nor our characterization of the necessary elements for a solution. But without exception, every single person there representing a potential demonstration site expressed strong reservations about using any “non-cots” solution in their organizations—even in a demonstration context. Not even two years earlier, these same companies had all stated to us and to Air Force ManTech management that the multiple BOM reconciliation problem was a major cost, schedule, and quality driver, and that a solution to that problem would be of great value to them and to the rest of industry. Now they were saying that such a solution would have value *only* if it was available from a software industry analog of KMart.⁹ Subsequent to this exasperating and disappointing

⁹ Air Force ManTech is not in the business of funding the development of commercial software. Their essential mission is to identify and cogently articulate industrial base capability voids vis-a-vis DoD weapon system acquisition and sustainment needs; define requirements for solutions to eliminate those voids, develop fundamental enabling technologies and critical elements of those solutions where required, where appropriate, and to the extent necessary to demonstrate feasibility; and, to incentivize industry through programmatic and other mechanisms to pursue development and deployment of those solutions. This is not to say ManTech is unconcerned with commercial viability—it is. Eventual commercialization is *one* tactic among many to disseminate new ManTech-sponsored technology and solutions.

In that vein, ManTech made it a MEREOS program requirement that we develop a comprehensive business strategy for PACIS technology commercialization. We initiated a strategic planning activity in early 1996 to satisfy that requirement, and by May of 1996 we had produced the first of several documents that would eventually become components of a documented Ontek business plan. This first document was an assessment of relevant environmental factors and trends, specifically focused on post-Cold War downsizing of the defense industrial base, the transformation of the software industry from a requirements-driven engineering-oriented industry to a market-driven commodity business, and on the implications—almost all negative—of those two factors for PACIS technology commercialization. It was quite a shock to see our predictions concerning attitudes of A&D contractors towards advanced information systems technology stated in that May document be completely validated—with a vengeance—the following September.

We presented the first version of the complete business plan to ManTech management in February 1997, and presented updated versions in September 1998 and March 1999.

episode, we attempted to overcome this Catch-22 in meetings with Deere, Caterpillar, and Lockheed Martin Aeronautical Systems in Georgia (LMAS), and to convince at least one of them to act as a Phase III demonstration site. We eventually succeeded at LMAS, *only* because we were able to show that MEREOS was *not* a commercial PDM or ERP system competitor, but was rather an “add-on module” or “plug-in” to those systems—a “structure server”—that is, a developer-oriented *tool*, *not* a stand-alone application system. In other words, if we had not had to shift our technical focus and make the requisite change in MEREOS software architecture, we would have not secured a Phase III demonstration site. Nor, to anticipate the subject of the third change described below, would we ever have been presented with the opportunity to participate in the re-design of a major prime contractor enterprise, or produced a major MEREOS program result—a baseline design for a new aerospace enterprise operating system.

The technical advantages gained as a consequence of the focus change were equally compelling if not more so. With the resources allocated to executing the abovementioned enhancements freed up, we able to bring a great deal of effort to bear on developing the formal systematics required to represent the abstract intentional structures underlying systems engineering processes and their products.

We had, among other things, demonstrated fully declarative and directly executable representations of processes in two prior versions of PACIS. Although many of these were very complicated,¹⁰ the structures involved were nowhere near the level of abstraction of those underlying systems engineering processes and their products. And it was precisely these structures that had to be explicitly represented if the increase in MEREOS representational amplitude was to be accomplished. Moreover, systems engineering is a formal process; however, most of the structures constituting its conceptual underpinnings have not, as yet, been formalized. That is, definitive and detailed descriptions of systems engineering processes and products were readily available, but suitably detailed formal characterizations of core systems engineering entity *types* were not.¹¹ Nevertheless, formal characterizations—technically, formal taxonomic delimitations—are essential raw materials for implementing computer-interpretable representations of *anything*, including these and other systems engineering entity types. We accordingly undertook to develop them, though this was not a trivial undertaking for two principal reasons. First, *all* of the principal conceptual structures underpinning systems engineering are, to varying degrees and in different guises, active subjects of current phenomenological and ontological research. In fact, the very notion that intentional structures are tractable to formalization *at all*, at least without eliminative reductions to the extensional structures of logic and set theory, is very recent and in some quarters is still vigorously disputed.¹² Second, we had begun a complete re-design of PACIS meta-level representation structures—PACIS *metasystematics*—just prior to MEREOS. Motivated by numerous lessons learned from our experiences

¹⁰ For example, to demonstrate subsumptive integration of foreign systems in PACIS during the Northrop/Air Force Automated Airframe Assembly Program (AAAP), we developed a complete declarative representation of the *entire* Ims database management system schematics—its representation structures, data definition, *and* data manipulation language operators—and we used this implementation metascheme as the basis for a PACIS subsystem that automatically generated representations of Ims database definitions, and another that generated-hoc queries against them during execution of PACIS process representations. Ims is notoriously complicated; as one Northrop database administrator and systems programmer once remarked: “If you can ‘do’ Ims, you can do *anything*.” Well, not quite: you can’t ‘do’ systems engineering requirements traceability management automation, for example...

¹¹ Examples of these are entity types such as REQUIREMENT; DERIVATION, ALLOCATION, and SYNTHESIS (the relations), and Measure of Effectiveness (MOE). This absence of formalization despite mature practice is analogous to the situation in mathematics. Mathematics as a formal enterprise is at least 2,400 years old, but systematic efforts to formalize NUMBER itself only began in the late 19th century. Among other things, efforts to formalize the foundations of mathematics led to the development of electronic computing machines. This analogy demonstrates that a great deal of both practical and theoretical work can be accomplished without reflective formalization, but implementing intentional processes in terms of computer programs can’t be.

¹² For that matter, phenomenology as a distinct philosophical discipline is itself less than 120 years old, and formal ontology is only 100. The criticisms of so-called “artificial intelligence” leveled by Hubert Drefus and his fellow-travelers are an example of one dimension of the controversy over the formalizability of intentional structures. Our favorite counter to certain of these arguments is a quip by John Searle: “Of *course* the background can be represented. Here goes: ‘the Background’.” [18.1] Although unhelpful from a software engineering perspective, it’s clever and funny.

with the prior version of the system, the aim of this effort was to put the metasytematics on new and completely self-applicative foundations, and to systematically re-derive the principal schematic elements of the representation system from those.¹³

Metasytematic structures are by definition the most abstract possible and self-applicative meta-structures are the most intricate; they are the elements of pure autothetic form. It is accordingly necessary from a practical standpoint to inform metasytematic efforts with concrete and suitably illustrative examples. It is also a formal necessity to partly evaluate the results of such efforts in terms of their capabilities to explicate these and any other possible exemplars. The conceptual structures of systems engineering were an excellent addition to the examples we had already been using for this purpose.¹⁴ Their intimate relationships to product and process structures meant they were germane to the other examples we were using to inform this endeavor. However the fact they are essentially intentional meant they were very different in kind from the others, and the need to systematically explicate these taxonomic 'gaps' in terms of metasytematic constructs represented a decidedly non-trivial 'stretch' for our efforts. But it was the differences between the three example sets that was actually, in the final analysis, the big payoff—or rather, the solution to the problem of developing the requisite metasytematics to subsumptively and systematically explicate these differences was. The solution to this problem, which we call the *structure abstraction* or *structure factorization* solution, represented a major technical advance achieved by the program, and, as we will see shortly, it provisioned us to take advantage of a major strategic opportunity as well.

None of this work was visible to any PACIS stakeholder except the formal team; it is the nature of abstractions to be invisible except through their concrete instances. However, the new metastructures greatly enhanced our understanding of the structures underlying systems engineering, systematics, and artifacts, and that deeper understanding was manifested to others in our abilities to clearly articulate the root causes of and the necessary characteristics of solutions to problems like the multiple BOM problem.

2.6.3 Change 3: MEREOS Phase III, Redefined

The third change was both a radical shift in focus and a major change in venue. It transformed MEREOS from an information systems program focused on tactical product representation problems into an enterprise architecture program, focused on the strategic issues involved in designing a new aerospace enterprise operating system.

We mentioned earlier that after lengthy discussions with several organizations, LMAS had agreed to serve as the MEREOS Phase III demonstration site. This agreement was reached in mid-1997. During 1998 we held a series of discussions with LMAS personnel in Marietta to solidify the details of this agreement, to select a particular program to host the Phase III demonstration, and to define specific objectives, mutual responsibilities, and qualification criteria for the demonstration system. Over the course of these discussions, our understanding of LMAS product definition and representation practices deepened, as did our familiarity with two related information systems implementation initiatives. Participating LMAS personnel also gained a comprehensive understanding of MEREOS technology and its relationships to those practices and initiatives.

Like any other classical aerospace prime, many LMAS processes and associated practices were divisionalized along programmatic lines. These included the Product Realization process, many of its subprocesses—notably Product Definition—and its related product representation practices.

¹³ The most critical metasytematic structures were the TAXON and CONCRESCENCE taxa and their primary noumenal derivatives OCCURRENT and CONTINUANT. The principal schematic elements of representation system are ENTITY and INTENTIONALITY. These are derivatives of the primary noumenal taxa, and they constitute the superstructure of the representation system.

¹⁴ The two example sets we had been using before this change occurred were complex product and process structures and those of biological systematics—the latter being the most well-developed and explanatorily powerful we know of. In fact, the PACIS metasytem is a generalization and self-applicative abstraction of biological systematics. It is a generalization because our representations must encompass more than living things. It is an abstraction in that such things as SIMILARITY, DELIMITATION, and IDENTIFICATION (and their numerous variants) are all taxa, and it is self-applicative in that TAXON is itself also a taxon. The relevant corners of the metasytem (the phylogenetic taxonomy of SIMILARITY and that of TAXON itself) are depicted by Figures 31 and 28 in Attachment 2, respectively.

Extreme differences in program age had also brought about even greater divergences between each program-specific variant of these processes, well beyond the norm. For example, c-130 is a very mature program; deliveries of the first production aircraft had been made in December 1956. The c-130 product definition is accordingly historically deep and systemically broad, at that time encompassing 42 years of engineering, countless variants, models, and options—many of which were still actively being produced—and it also included the c-130J: “an entirely new airplane.” In contrast, F-22 was still in pre-first flight EMD. Unsurprisingly, c-130 Product Realization processes—especially c-130 Product Definition and engineering release processes—had very little in common with their F-22 counterparts beyond superficial similarities due to shared materiel, finance, and HR functions. Most importantly to us, their product representation practices and product data conventions differed dramatically, and so did the information systems each program used.

An initiative to establish a common Product Definition process across programmatic lines had been underway for some time prior to our engagement with LMAS. Impelled by looming F-22 and c-130 J production program cost reduction needs, this initiative had adopted the approach of creating a common process via use of a commercial PDM software package. In early 1998 a second initiative to establish common Product Delivery and supporting infrastructure processes had also been formally launched. Also impelled by F-22 and c-130J program needs, this initiative was focused on creating commonality via use of a commercial ERP software package. These two initiatives were similar in several respects. Both were mandates of Lockheed Martin Aeronautics Sector headquarters; LMAS was their first target implementation site (the other sector companies were to follow); each was predicated on an explicit “cors only” policy; and neither was going to be capable of addressing real LMAS product representation needs—although this latter point was only understood by a small cadre of LMAS cognoscenti, and was not generally recognized by either of the teams leading these two initiatives or by LMAS and Aeronautics Sector management.

In light of this situation, two facts had become increasingly clear at this stage of our discussions with LMAS. The first was that demonstrating the *MEREOS solution* to complex aerospace product representation needs and requirements would *not* constitute much value for LMAS. The second was that exploiting our command of the relevant product representation *requirements* to support their enterprise process improvement and related information systems implementation initiatives *would* constitute value for LMAS.

To help address the product representation capability shortcomings of these initiatives, and to contend with the impedances to the *MEREOS* Phase III demonstration objectives presented by them, we, together with our internal LMAS sponsors, undertook a vigorous and concerted effort to bring the relevant *requirements* to the attention of all concerned. To accomplish this we conducted a series of presentations to demonstrate the specifics of complex product representation needs and requirements to both teams; we wrote and distributed documents outlining these; and we demonstrated software to illustrate key capabilities that any candidate solution offered by either of these initiatives would necessarily have to provide. Thus all throughout 1998, we played the role of Product Definition and Product Representation *process architects*, emphasizing our command of the relevant requirements, and de-emphasizing our capabilities to produce information system solutions to those requirements, except as a vehicle for demonstrating the feasibility of developing systems to provide the requisite capabilities.

During 1998 we had also taken on an architectural mantle in another LMAS initiative as well, putatively unrelated to the Sector-mandated PDM and ERP initiatives. LMAS leadership had started an enterprise productivity (EP) initiative in mid-1997, motivated both by programmatic and by strategic needs to enhance LMAS competitiveness by improving enterprise process efficiencies and overall effectiveness. Cognizant of our prior work on similar programs in other companies, the leader of the LMAS EP initiative brought us in to assist in developing a strategic plan for governing and executing the LMAS EP program. We produced two substantive contributions to the program as a result of this effort. First, we developed a formal strategic plan for the EP program which was explicitly linked and thus directly traceable to both the LMAS company and Aeronautics Sector strategic business plans. The principal tactical element of that strategic plan was a program to develop and deploy a new LMAS enterprise architecture, and the second product we produced was a detailed plan for that program.

Both of these activities were heavily dependent, albeit in different ways, on the metasytematics we had developed to support MEREOS representations of the abstract intentional structures underlying systems engineering processes and their products. In fact, we would not have been able to execute either of these architectural efforts without first having had these results in hand. The Product Definition and Representation needs and requirements analyses we presented to the PDM and ERP teams were developed using our formalization of systems engineering methods, and they were framed in terms of those structures. The formal scheme used to define the EP strategic plan and its explicit traceability to the LMAS and Sector strategies was a direct result of our applying a generalization of systems engineering methods to strategy development. And most significantly, our approach to developing the LMAS enterprise architecture was predicated on the use of new formal methods, based on the synthesis of biological systematics and systems engineering we had developed as a result of our effort to re-design the PACIS metasystem.

It had also become obvious to the most casual of observers as a result of our activities during 1998 that the original objectives for MEREOS Phase III were going to have to be re-structured to accommodate LMAS realities. Then a truly fortuitous event occurred: a new LMAS president was appointed in early 1999, and his personal mission was to “re-invent” LMAS. The opportunities presented by this leadership change to all MEREOS program stakeholders were utterly clear. The new president was briefed on the proposed enterprise architecture program and our envisioned role in it in early March. We consulted with cognizant Air Force personnel to discuss our participation in this program under MEREOS Phase III auspices. LMAS agreed to match Air Force funding to support the changes and increases in Phase III tasking, and in May of 1999 the LMAS Enterprise Architecture Program (LEAP) was officially launched, fully supported by an Ontek team. This last and most far-reaching of the major changes marking the evolution of the MEREOS program had occurred.

The ship wherein Theseus and the youth of Athens returned had thirty oars, and was preserved by the Athenians down even to the time of Demetrius Phalereus, for they took away the old planks as they decayed, putting in new and stronger timber in their place, insomuch that this ship became a standing example among the philosophers, for the logical question of things that grow; one side holding that the ship remained the same, and the other contending that it was not the same.

Plutarch, *The Life of Theseus*

3

RESULTS PART I: *Product Structure Definition and Management*

The two primary MEREOS program objectives were first, to identify the root causes of the multiple BOM reconciliation problem and define the requirements for a solution, and second, to prove, via demonstration, the feasibility of developing a system to implement that solution. Both of these objectives were achieved. That is, we produced a rigorous and detailed definition of formal requirements for a product data management system—one capable of addressing the *real* needs of both aerospace and defense contractors and other manufacturing enterprises that produce complex, technically advanced products. We also developed and demonstrated software that proved the feasibility of creating a system that satisfies those requirements, thereby also achieving our aim of demonstrating the value of PACIS technology in a strategically visible business context.

But success defined in terms of achieving programmatic objectives is not necessarily coincident with delivering value to stakeholders. Nor is the value realized by achieving one objective necessarily equal to that of the others. In fact, for reasons we will briefly sketch now, the formal requirements definition we produced is of much greater potential value to industry—and thus to ManTech and DoD—than the software we built and demonstrated.

Consider the following facts. Every major prime contractor has developed and implemented at least one product data management system. Most have created and used several. Many “commercial-off-the-shelf” (COTS) PDM systems have been available for some time. Every major prime contractor has licensed and is using at least one of those as well. Despite this abundance however, familiar and long-standing product representation problems continue to hinder industry efforts to lower costs, improve quality, and reduce cycle time. The multiple BOM reconciliation problem is simply one illustrative example of these problems. Given these facts, it should be obvious that what aerospace and defense contractors and their commercial peers *do not* need is yet another PDM system. What they *do* need is an effective, implementable, sustainable, and *total* solution to *all* of the requirements deriving from their product representation needs. And it is also a fact that no such solution exists,

notwithstanding the claims of some software vendors to the contrary. Moreover, a PDM system will *never* constitute a complete solution to product representation requirements: regardless of its capabilities, an information system is only *one* element among many necessary to realize a total solution to those requirements.¹⁵ Thus even if a system with the requisite capabilities to address real product representation needs existed—and it doesn't—or even if the MEREOS program had produced a commercially available production version of such a system—and this was not a MEREOS program objective—such a system would only constitute a partial solution.

Several factors are impeding development of a truly effective, albeit partial, information system solution. These in combination with others are inhibiting development of a total solution. None are mere programming problems. Nor for that matter are they ultimately methodological, organizational, or even strategic issues, although many of these are symptomatic of the underlying root causes. The truly operant impediments are *conceptual*—they are *formal process* problems. Specifically, they are Product Representation and superordinate Product Realization architecture problems. To be precise, the two greatest obstacles blocking development of both a total enterprise solution and an effective information system solution are the absence of cogent requirements definitions and the consequent lack of implementable balanced designs for these processes.

In the light of these facts, it should be clear why our development of an experimental PDM application to demonstrate an entirely new kind of information system, no matter how powerful, represents little strategic value to external MEREOS program stakeholders. It should be equally clear why a comprehensive needs analysis and systematic requirements definition for the Product Representation process *does* constitute significant value to those stakeholders—it is impossible to provision a poorly understood process with an effective information system, and it is counterproductive to automate the data creation and management activities of a flawed one.

Real complex product representation needs primarily consist in structure multiplicity, other specific dimensions of representational amplitude, and concomitant correspondence and integrity. The critical requirements comprise 17 specific metastructures, 3 classes of functional capabilities, and 2 primary supervening organizational and enterprise process requirements. As far as we have been able to ascertain, these results constitute the first formal systematic explication of complex product representation needs and requirements ever produced that is explicitly focused on meta-level structures and situated in a total enterprise operating system context. Because of its intrinsic value to industry, and thus ultimately to ManTech and DoD, this needs analysis and requirements definition is the focus of our presentation here.

3.1 Product Representation Needs

Product Realization is the primary enterprise mechanism for delivering customer value and is, accordingly, the core process of any manufacturing enterprise. This process constitutes a technical and customer-centric axis of SOLUTION REALIZATION, one of the three major enterprise processes, and it is aligned with the product life cycle in ways we only touch on here. The primary external inputs to Product Realization are technical customer needs. Its principal outputs are product/process systems exemplifying the performance and effectiveness characteristics required to satisfy those needs. As depicted in the figure on the following page, there are three major Product Realization subprocesses: *Product Definition*, *Development* (sometimes called Production or Delivery), and *Support*. The first two of these directly constitute phases in the product life cycle. Product Support does not,¹⁶ although this does not impact the product structure definition and management issues that concern us here.

¹⁵ Other notable elements of a total product representation solution are suitably capable Product Representation and Realization *processes*, properly aligned organizations to optimally execute and govern those processes, and requisite infrastructure for provisioning them both. These and other crucial enterprise operating system elements are discussed in Section 4: Results Part II.

¹⁶ At least as classically understood and as executed by manufacturing enterprises. Product Support is a logistics-centric *cut* through *customer* processes involving use of manufacturing enterprise products. INTERDICTION [using an F-16] and HARVESTING [using a Deere 9750 STS combine] are examples of customer processes that can be instrumentalized by these products. *Spareing*, which is classically executed by the producing manufacturing enterprise, is a prosaic example of a logistics-centric cut through both of these.

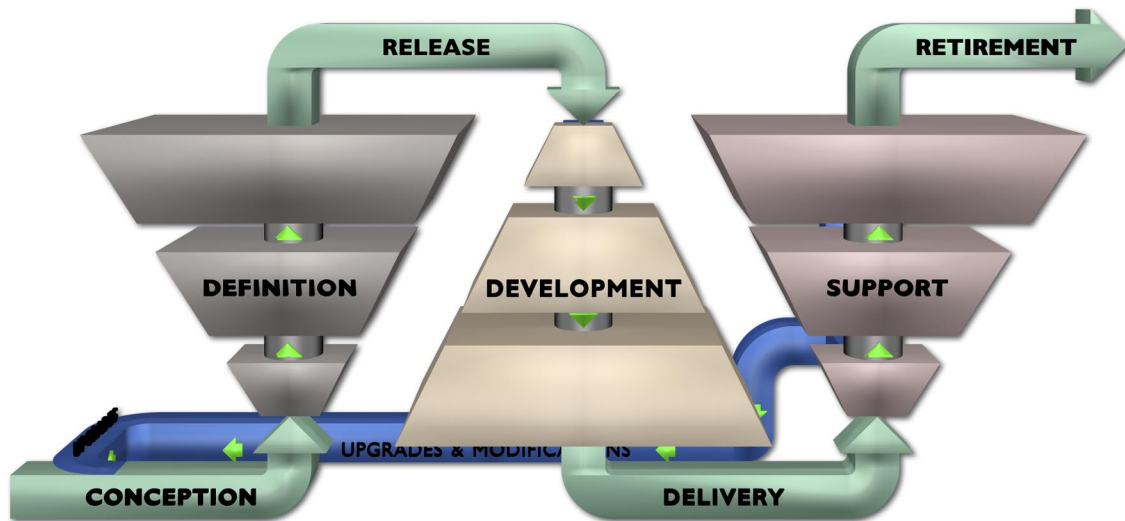


Figure 2. Product Realization Process Elements

The principal inputs to Product Definition are instrumentality capability voids—technical needs. Its primary outputs are complete prescriptive system delimitations—‘total definitions’—verifiably representing technical solutions to the requirements deriving from those needs. The principal inputs to Product Development are descriptive and executable elements of product definitions called “build packages.” Its primary outputs are deliverable systems demonstrably realizing those definitions and related documentation. Aside from the deployed products themselves, Product Support inputs are diverse but mainly comprise logistics (spares, test equipment, etc.) and product definition elements collectively called “tech pubs.” These latter generally fall into two groups: descriptions of product operations and executable sustainment definitions. Its principal outputs are product systems in states of operational readiness (including requisite provisioning), improvement requests and defect identifications, and, ultimately, product system retirements or disposals.

Even these superficial characterizations should make it clear that product data are the primary information assets required to provision, execute, and govern the Product Realization process. All Product Realization subprocesses and activities both require and generate specific types of product data. Moreover, these data are utterly pervasive. With few exceptions,¹⁷ *all* manufacturing enterprise processes and activities depend on product-related data in some way, albeit to differing degrees, and all generate product-related data, though of diverse and in some cases only indirectly connected kinds. Thus the efficiency and effectiveness of all enterprise processes are intricately but nonetheless directly related to product data completeness, fidelity, and availability—and, consequently, so is the strategic vitality of these enterprises. Given the nature of manufacturing and the central role of the Product Realization process in such enterprises, this is hardly surprising.

Product *definitions* comprise both the integrating superstructure for product data and the schematic underpinnings of all other enterprise data—they are the conceptual cornerstones of the manufacturing enterprise. Representing product and other related entity *types*, their structures, variants, attributes, and relationships among these, product definitions are the premier information assets of any manufacturing enterprise. For organizations that create complex technically advanced products, they are also major strategic assets, and for aerospace and defense contractors, they are capital deliverable products as well, requiring years of effort by legions of skilled people to produce, and substantial infrastructures to maintain. Engineering defects—flaws or deficiencies in product definition *content*—have serious operational, economic, and hence strategic consequences for product users and for the manufacturing enterprises that produce them. Formal representation defects—flaws or deficiencies in product definition *informatics*—also have severe impacts on Product Realization process performance and effectiveness. Unlike engineering defects, informatic deficiencies are

¹⁷ Certain extrinsic regulatory compliance activities such as EEOC reporting would be one example.

invisible to product users and only obliquely visible to producers, as they tend to be obscured by the very visible and negative effects of several pervasive Product Realization problems. Engineering release, logistics integration (so-called “supply chain integration”) and engineering change propagation are three notable and inter-related examples of these problems. Informatic deficiencies lie at the center of them all, and many others besides.

Thus even at this shallow level of analysis, two intrinsic elements of Product Realization immediately stand out as critical enablers of enduring superior capability. The first is the process of producing and sustaining product definitions—call that the *Systems Engineering* process. The second is the process of producing and sustaining the informatic metastructures and systems that *frame* product definitions, which we will call the *Product Representation* process.

As we conceive of it here, the Systems Engineering process is Product Definition. More precisely, it is a specific configuration of that process tailored for rigorously defining complex products that typically require substantial capital investment and novel technology to develop, and extensive infrastructure to use and sustain, usually over decades of service life. But Systems Engineering is, nevertheless, the Definition subprocess of the Product Realization process. The Product Representation process is *not*, in contrast, a Product Realization subprocess—it is, rather, a particular *variant* of that *entire process*. That is, invoking the context of enterprise architectures, Representation is the Product Realization process *itself*, applied to informatics.¹⁸ Its principal products are *information systems*—instrumentalities for provisioning decision-making processes; tools of purposeful thought. Product Representation is simply Representation applied to *product* informatics, and *its* information systems products are tools for provisioning intentional activities involved in the Product Realization process, including—but certainly not limited to—Systems Engineering analysis, synthesis, and evaluation.

A great deal of focused expertise has been brought to bear over the years to define Product Realization as a formal process. The Systems Engineering process in particular is very well understood and thoroughly codified, as the large body of good quality specifications and textbooks describing it demonstrates. On the other hand, understanding of Product Representation as a formal process in its own right is not mature, although several efforts to ameliorate this have yielded considerable progress and have produced significant benefits for industry.¹⁹ But the continued existence of problems such as multiple BOM reconciliation are symptomatic of the fact that at least some core product representation needs have not yet been cogently identified and articulated—a least not to the extent necessary to enable viable solutions to these problems to be developed.²⁰ We will now turn our attention to these needs.

¹⁸ Actually, Representation is *Solution* Realization applied to informatics; and is therefore a variant of that process as is Product Realization. However the systematic complexities of the distinctions between these two processes need not concern us here.

¹⁹ PDES/STEP [13] is a notable example. This family of standardization efforts has produced great deal of quality work aimed at standardizing product definition data to facilitate its exchange among differing information systems.

²⁰ To be fair, many product representation problems sit directly on the shoulders of long-standing and elusive ontological problems; it is not surprising some of these are so resistant to solution. For example, the nature of Part-Whole and Dependence relations (Assembly→Component and Part→Material relations in BOMs) were studied in detail by Aristotle, but *formalizations* of these useful for producing product representation systems were only developed very recently [20]. The nature of change, substance, and the status of artifacts are subjects of long-standing philosophical debate, as is illustrated by the quote from Plutarch at the beginning of this section. Engineering change impact identification, propagation, and traceability management problems are directly related to these questions. There is in fact a much closer relationship between product representation problems and some protracted philosophical issues than one might initially suspect, and some philosophical work to address these (though not as much as one would hope) is directly relevant and helpful. It is certainly possible to build an PDM system without understanding the intricacies of these and other related issues; people have done so for years. It is *not* possible to produce one that solves the problems we are concerned with here without such an understanding *and* an ability to encode that understanding in a programming language. You can't hand-wave a C++ compiler; you can't represent something you don't know exists in a STRUCT statement, and you can't automate a process you don't understand in C++ code.

3.1.1 Structure Multiplicity

Product definitions are *representations* of artifact *types*. Artifacts are entities which are designed and created by certain specific intentional processes, or which are instrumentalities used by intentional agents to execute processes, or both. Types are *taxa*; groups of related entities. Representations are artifacts of intentional processes that objectify—i.e., present—other entities to intentional agents. Many kinds of product data are representations of artifact taxa. Some represent individual members of artifact taxa. Process architectures depicted by functional flow diagrams and correlated synthesis architectures represented by E-BOMS, M-BOMS, and L-BOMS are common examples of the former. Individual physical architectures represented by serialized “AS-BUILT” and “AS-REPAIRED” BOMS are examples of the latter. Both kinds are of interest to us here.

It is an empirical matter of fact that a given product can, and almost invariably does, actually instantiate multiple coincident and divergent structural configurations. That is, there is no such thing as “the” structure of an artifact type or even of a particular individual artifact, if by the term “the” one means a single incontinent and invariant structure. It is a formal matter of fact that the origins of many divergences among these configurations are contrasts among relations between artifacts and concrete processes. It is consequently a pragmatic matter of fact that the differences between these exist for both formal and practical reasons. Hence it is an analytic fact that these classes of divergence cannot be explicated in terms of mere representational conventions, or differences among ‘views.’ And finally, it is an effective fact that they also comprise one major root cause of product realization and representation problems—most notably the steep costs, long lead times, and high error rates associated with propagating engineering change in complex manufacturing environments. We will support and illustrate these arguments by first enumerating some representative types of structural configurations, and then follow that with some specific examples of differences among them. Note however that the phenomenon of structure multiplicity is much broader and more complicated than is conveyed by these specific examples, and, accordingly, so are its representational consequences and the corresponding need.

3.1.1.1 Representative Structure Types

All complex product systems exemplify at least four distinct and specific structural configurations: *nominal*, *constructive*, *sustainment*, and *effective* configurations. Each of these differ from the others both in form and in content. Effective configurations are determined by application contexts and are, accordingly, related to processes in which product systems are used in instrumental capacities to execute these processes. The first three configurations are correlated to and conditioned by product life cycle phases, and are, accordingly, related to major Product Realization subprocesses in ways we will describe in more detail below.

■ Nominal Configurations and E-BOMS

The E-BOM for a product system such as the F-22 represents its abstract physical architecture. This architecture defines an instrumentality for realizing the *operational axis* of the F-22 functional architecture and its attendant performance and effectiveness requirements. It is produced by a systems engineering process called *synthesis*, and the physical architecture it represents reflects a ‘top-down’ focus on synthesizing these operational requirements. Accordingly, all elements constituting that E-BOM must be *traceable* to—meaning they are verifiable solutions to—at least one of these requirements. Thus every particular F-22 *must* veridically instantiate the structure represented by the F-22 E-BOM; if it doesn’t, it is *defective*—it is not a valid material realization of that axis of the functional architecture.²¹ Should it be flown in that condition, it likely would not be capable of executing its mission with the requisite degrees of performance and effectiveness.

²¹ Lockheed won’t get paid if it doesn’t either. The customer pays based on acceptable integration and flight test results, and thus pays indirectly off the E-BOM.

I Sustainment Configurations and L-BOMs

The L-BOM for a product system represents its abstract infrastructural architecture. This architecture defines an instrumentality for realizing the *logistic* axis of the functional architecture for the product system and its attendant reliability and supportability requirements. Like an E-BOM, an L-BOM is also a product of the synthesis process, and the physical architecture it represents reflects a concomitant 'top-down' focus on synthesizing these integral logistic requirements. Accordingly, all elements constituting an L-BOM must be traceable to at least one of these requirements, and every *deployed* product *must* veridically instantiate the structure represented by its L-BOM. If fails to do so, it is *deficient*—it is not a valid material realization of that axis of the functional architecture. Should it be used in that condition, it likely could not be maintained in the necessary state of readiness to execute its mission with the requisite degree of reliability.

I Constructive Configurations and M-BOMs

An M-BOM for a product system represents a specific *realization* architecture. This architecture is the structural complement of a specific and *qualified* production process architecture and its attendant cost, schedule, and quality requirements. Both this process and its correlate M-BOM are produced by a process typically called *manufacturing engineering*, and the specific structures they represent reflect a 'bottom-up' synthesis of realization requirements required to actually implement the abstract physical architectures represented by the E-BOM and L-BOM for a given product system. Accordingly, all M-BOM elements must be traceable to at least one E-BOM or L-BOM element, *and* be traceable to requirements reflecting producibility, availability, socio-economic, and regulatory constraints. Thus every particular F-22 must veridically instantiate the structure represented by the F-22 M-BOM; If it doesn't, it is *invalid*—it was not verifiably produced in accordance with a qualified production process. Should it be accepted in that condition, it may not be possible to validate its traceability to its specified abstract physical architectures, and, therefore, to its functional architecture, rendering both its requisite operational status and logistical qualifications indeterminate.

I Derivative Configurations

The configurations described above comprise actual matters of structural fact correlate to a specific processual architecture—albeit diverse and in some cases only contingently or periodically operant—and, consequently, their respective BOMs are intended to veridically represent those particular structures. Together with a few others we did not enumerate here, these foundational or *genitive* configurations collectively delineate the class of structures connected with creating, using, and sustaining complex technical systems.

A second class of configurations also exists, over and above genitive configurations. Created from and supervening on these, *derivative* configurations comprise *analytic* matters of structural fact keyed to specific and purely *intentional* operations involved in creating, using, and sustaining complex systems. Developed to facilitate the execution of these processes by selectively abstracting from the structures of other configurations, derivative configurations are also associated with specific representational constructs specifically designed to convey their particular analytic structures. Comprising redactions of other structural configurations, derivatives are idealizations, and unlike the representations of genitive configurations, their representations *are* views.

The most illustrative examples of derivative configurations are those used in resource and production planning, scheduling, and cost accounting operations. Created by abstracting from nominal and constructive configurations, these configurations are represented by various special-purpose "pseudo" BOMs, such as "modular," "phantom," and "flattened" or "matrix" BOMs.²² Logistical analogs of these exist as well, comprising redactions of nominal and sustainment configurations. Being artifacts of analysis explicitly designed to facilitate subsidiary intentional processes, derivative configurations of course differ substantially from the actual matters of structural fact from which they

²² Matrix BOMs for instance are used to compute requirements for components that are common elements of several similar products or product families; the underlying configuration represented by these contains only those elements with two or more "parents"—so called "multiple where-used" and "make multiple from" elements. The APICS dictionary [2] contains definitions and brief descriptions of these and other similar structures

are abstracted. As we are primarily concerned with divergences among the actual structural facts themselves—major factors in cost, lead time, and error rate attributes of product realization and representation processes—derivative configurations and their attendant representations are only of peripheral of interest to us here.

3.1.1.2 Representative Structure Divergences

As we stated above, every particular product system will necessarily instantiate several genitive structural configurations, specifically including a nominal, sustainment, and constructive configuration. And, each of these configurations will inevitably differ from the others and so, accordingly, will the E-BOM, M-BOM, and L-BOM respectively representing them. In light of the above characterizations, neither the fact these configurations differ nor the fact a particular system necessarily exemplifies them all should come as much of a surprise. Each of these configurations is a synthesis of a specific processual architecture and set of requirements—all of which must be satisfied by a given product if it is to attain the requisite degrees of performance, effectiveness, reliability, and validity.

Divergences between two distinct genitive configurations of a particular complex product system—for instance its nominal and constructive configurations—exhibit definite patterns, although the individual intricacy and scale of each individual configuration can make it difficult to discern them. Internal variability, contingency, and changes to an individual configuration over time can also obscure divergences between that configuration and its counterparts. But while the multi-faceted phenomenon of conditionality is a hallmark characteristic of complex technical product systems, especially within nominal and effective configurations, it is neither the genesis of nor a basis for explicating divergence between *pairs* of these configurations.²³

Pair-wise divergences between genitive configurations are *positional optimization contrasts*—they are *products* of deliberate efforts to optimize the *relationships* that artifacts stand in to the processes that define, create, sustain, use, and terminate them, governed by formal distinctions between these relations themselves. Thus the inter-configuration divergences of interest to us here are not only inevitable: they are, in fact, signatures of design expertise.

■ Articulation

An atomic element in one product structure configuration can frequently be a complex or composite element in another. We call this form of divergence *articulation*, and it is a common cause of differences between BOMs. For instance, the left and right wing spars of the F-18 are syntheses of certain aerodynamic load-bearing and rigidity requirements, are consequently defined as atomic parts in the nominal air vehicle configuration, and are therefore represented as such by the F-18 E-BOM.

However, each of these spars are actually constructed by first machining and then welding several component parts together. That is, it is economically unfeasible to procure the required size of aluminum bar that is also free of internal voids. Moreover, it is a practical impossibility to install single-piece spars during assembly, as there simply isn't enough maneuvering room; the spars must be created by welding several previously machined parts together after these are installed 'in rig.' Thus each spar in the *constructive* air vehicle configuration is an *assembly* rather than an atomic part, and it is accordingly represented as a subassembly on the F-18 M-BOM.

²³ Except in the most trivial senses. Conditionality within the configurations of given product system *type*, such as c-130, is one genesis of divergence between *all* of those configurations *collectively* and the configurations of the concrete *instances* of that type—i.e., an individual c-130. For example, there is certainly nothing variable or contingent about the forward and aft extensions ('plugs') in the fuselage of a particular c-130 belonging to the British Royal Air Force, nor about their absence in one equipped for short takeoffs belonging to the U.N., and naturally, their respective "as-built" and "as-maintained" configurations will differ substantially. However, differences such as these are traceable to conditionalities in effective, nominal, sustainment, and constructive configurations known as *model effectivities*. These define elements that are operant ("effective") within the scope of a specific variant of those configurations, but which are *inoperant* ("not effective") outside that scope. Hence any two particular instances of these models will of course be structurally distinct. A major genesis of differences between a particular c-130H built in 1985 and a new c-130J is *versioning*—another variant of conditionality—and these are traceable to version effectivities within the configurations of given product system type.

Articulating an element in one configuration into several elements in another can be done in several ways and for a variety of reasons beyond those of economic efficiency and assembly constraints in realization process contexts. That is, articulation divergences are not limited to nominal/constructive configuration pairs and hence are not limited to E-BOM/M-BOM pairs. Sound instances of articulation in product structure definition contexts always constitute solutions to specific engineering, materials, manufacturing, or logistical problems. They cannot be dismissed as bad practice or mere artifacts of representational convenience or eliminated by policy. Inter-configuration element articulations are necessitated by empirical matters of fact or by constraints deriving from one or more requirements.

■ Factoring

Two or more separate and diverse elements in a given product structure configuration can frequently be multiplexed realizations of a single element in another. We call this form of divergence *factoring*, and it is another common cause of differences between BOMS.

Some parts of complex artifacts inevitably have one or more complex and difficult to produce features, such as 5-axis surfaces, long holes with small diameters and tight perpendicularity tolerances, and helical gear teeth, to name a few. Manufacturing engineers commonly factor these into what are sometimes called *phantom* parts—the motive being efficiency and uniformity. These are gained by producing the feature once on the phantom part, subsequently splitting or dividing the phantom to derive the desired end products. The geometric forms of these kinds of phantom parts are determined almost entirely by the geometry of the feature to be produced, and their spatial extents are determined by the types and number of actual parts the phantom is designed to yield.

For example, helical gears in transmissions are syntheses of mechanical power propagation requirements, are consequently defined as individual atomic parts in a nominal transmission configuration, and are therefore represented as such by the relevant E-BOM. However, in the event such gears might share the same tooth geometry and raw material, any competent manufacturing engineer would factor these into a single blank, thereby enabling *several* similar gears in the nominal configuration to be produced from one part in the constructive configuration—that is, by hobbing the blank and then parting off the desired gears. Thus the separate individual gears would be elements in both the transmission E-BOM and M-BOM. However, the M-BOM would differ from the E-BOM, in that it would contain the gear blank as an element, while the E-BOM would not.

Like articulation, factoring separate and individual elements in one configuration into derivatives of a single progenitor in another configuration can be accomplished in a variety of ways. And, like those originating from articulations, divergences due to factoring can exist between any pair of configurations, and consequently between any pair of BOMS. Nor should these differences be denigrated. Quite the contrary: factoring is a powerful efficiency and effectiveness enhancement technique—multiplexing is a signature technique of expert design.

■ Consolidation

Two or more separate and diverse elements in a given product structure configuration can be a single unified element in another. We call this form of divergence *consolidation*, and it is a third cause of differences between BOMS. For example, each of the individual elements constituting the F-18 avionics subsystem are syntheses of various flight control and vehicle management requirements, are consequently defined as subassemblies and atomic parts in the nominal and constructive air vehicle configurations, and are therefore represented as such by the F-18 E-BOM and M-BOM.

However, many of these elements are actually deployed, inventoried, and replaced as integral units (“Line Replaceable Units” or LRUs) rather than as subassemblies of individually accessible components. Electronic subsystem elements such as motherboards or sealed electro-optical inertial guidance packages are paradigm examples of LRUs. In some cases LRUs are designed to be disposed of when any of their elements fail or preventative maintenance schedules call for their periodic replacement; in others they are designed so they can be disassembled, repaired and refurbished, reassembled, and subsequently put back into spares inventories.

Consolidating elements in one configuration into a single integral element in another can be effected in several ways. That is, consolidation divergences are not limited to nominal/sustainment

configuration pairs and hence are not limited to E-BOM/L-BOM pairs. Sound instances of consolidation in product structure definition contexts always constitute solutions to specific reliability, supportability, maintainability, and logistic availability requirements. And just like articulations, they cannot be ignored or eliminated by policy; inter-configuration element consolidations are mandated by empirical matters of fact or by constraints deriving from one or more requirements.

3.1.2 Amplitude

The structure multiplicity need just described is actually the specific *mereological*²⁴ variant of a more general need which we call *representational amplitude* or scope. In other words, the capability to explicitly represent multiple coincident and divergent genitive configurations of a given product system is *one* variation among several constituting a major class of amplitude needs, and many of the shortcomings of existing product representation systems are symptomatic of deficiencies in this and other variants of representational scope. We presented mereological amplitude (“structure multiplicity”) as a need distinct from and prior to the other variants because of its direct relationship to the multiple BOM problem. Descriptions of the others, no less crucial despite their indirect relationships to that particular problem, follow our remaining introductory remarks.

All representations, including BOMs, *are artifacts*—they are *products* of processes and *instrumentalities* for executing them. In these two specific ways they are indistinguishable from all other types of artifacts, and like all the others, representations are created, used, and sustained in order to address certain specific needs. Despite this positional equivalence however, representations as a class differ from other classes of artifacts in one crucial respect—they designate, describe, or depict—in some way or another, they *objectify*. That is, representations are instrumentalities of presentation, and it is this inherently revelatory characteristic that distinguishes representations from any other type of artifact. The same point conversely expressed is that representations exist to address a unique class of *informatic* needs, and the attributes constituting sufficiency conditions for satisfaction of those specific sorts of needs—i.e., *representational* performance and effectiveness criteria—are also correspondingly unique.

Three signature characteristics of complex technical products and their life cycles are complexity, variation, and novelty. Each of these exemplifies many different forms. For example, external or extrinsic complexity is *scale*. Internal or intrinsic complexity is *intricacy*. Processual or ontogenetic complexity is *extended interdependence*. Extrinsic variation is *heterogeneity*. Intrinsic variation is *contingency*. Processual variation is *change*. Extrinsic novelty is *instatement* or *rescission*. Intrinsic novelty is *consolidation*, *factorization*, or *separation*. Processual novelty is *innovation*. Thus extreme multi-dimensional diversity over comparatively long life cycles is a hallmark characteristic of complex product systems. Commensurate representational *amplitude*—the capability to coherently encompass the full spectrum of these diversities across entire product life cycles—is a principal and as yet unfulfilled product representation need that includes, but is not limited to, the mereological ‘dimension’ of amplitude previously described.

Scope of diversity is *delimitative*, not quantitative; it is a classificatory magnitude rather than a numeric one. Determining the sufficiency conditions for satisfaction of this need is consequently a *taxonomic* rather than a mathematical task. That is, stipulating that this need consists representations of 42 types of entities is meaningless; one must identify *which* types constitute the requisite representational scope—although one may of course count them after they have been identified if one wishes to. Note that in the interest of brevity the listing that follows consists in taxonomically related *groups* of types delimiting the requisite dimensions of amplitude, instead of particular types constituting those groups, or ‘locations’ on these dimensions.

■ Ontogenetic Amplitude —Representation of Processes and Other Occurrents

All product systems are both directly and indirectly correlated with processes in a myriad of specific ways. For example, most products instrumentalize (i.e., materially enable) the execution of processes; some directly execute them. The F-22 and the interception process is an example of the former; a database management system and data storage and retrieval processes is an example of the

²⁴ Centered on relations of part and whole—i.e., configuration or structure in the assembly/component relation sense.

latter. All artifacts are outputs of intentionally directed realization processes; many are inputs to these as well. Particular configurations of a product are keyed to specific application processes or realization process phases as we previously described in some detail. Finally, particular *versions* of product configurations are correlated with *changes* in application or realization processes. Application process changes are consequents of changes in instrumental needs or requirements resulting from shifts in user purposes. Realization process changes are made to correct flaws, mitigate deficiencies, exploit new technologies, and typically result in enhanced product performance and effectiveness.

Moreover, many ‘products’ of so-called *service* enterprises or service organizations within enterprises *are processes*, not products in the sense we have been using that term here. The formally *qualified* analysis, production, and assurance processes used by aerospace and defense enterprises and their commercial peers to create, validate, and support their products are examples of these, as are most of the ‘products’ of medical, financial, and governmental institutions.²⁵ All such processes are *themselves* artifacts, designed, developed and sustained via sophisticated realization processes like any other artifact; they also encompass effective, nominal, constructive, and sustainment configurations, and they are frequently comparable in complexity, variation, and novelty to the most complicated technical product systems.

The fundamental distinction between enduring objects that exemplify persistent qualities on the one hand and intrinsically dynamic entities such as processes and events on the other is utterly entrenched in our everyday conception of the world. Commonly called the *continuant/occurrent* distinction in philosophical circles, it is reflected in the grammatical forms of all of our natural languages by the distinction between noun and verb, and it is instantiated in various forms by almost all of the formal languages and systems we use to model the world and our representations of it. Furthermore, almost all of our languages and formal systems also exemplify a pervasive although implicit bias towards continuants, and nowhere is this more evident than in existing product representation systems, which without exception utterly fail to explicitly represent occurrents at all—let alone represent their structures—except in trivial textual form. Continuants *are* ‘objects;’ thus they are easily objectified for these purposes. However, occurrents are not ‘objects;’ one simply doesn’t put one’s hands on processes, nor easily determine their characteristics, except obliquely in terms of their outputs. Accordingly it takes a certain turn of mind to seriously undertake to *explicitly* represent them as entities *on par* with continuants, instead of implicitly and indirectly representing them via continuant surrogates—for example, as lines of text in documents.

The crucial need for representational *parity* across *both* sides of the continuant/occurrent divide should be clear, given the numerous and multi-faceted relationships between them. The imperative need for ‘product’ representation systems to explicitly represent occurrents directly should be equally clear, in light of their status as artifacts or ‘products’ themselves.

■ Nomological Amplitude —Representation of Needs, Requirements, and Metrics

All artifacts are created to instrumentalize the accomplishment of certain purposes. Purposes are possible states that one or more agencies desire to be actual; thus accomplishment is *actualization*—an outcome of *effective* actions or processes. “Effective” means that the result state is *satisfactorily congruent* to the desired state. “Satisfactorily congruent” means that some requisite *degree* or *measure* of effectiveness has been defined as a criterion of success.

Conditions that impede or block the execution of effective actions or processes are called *needs*—capability voids with respect to actualizations of desiderative states. Instrumental or intrinsic needs are capability voids respecting *mechanisms* of actions or processes—they are exigencies of artifacts. Thus satisfaction of these needs is *instrumentalization*—an outcome of effective artifact (‘product’) *realization*. “Effective” in this context means that the resulting artifact is suitably functional for the action or process. “Suitably functional” means that one or more measures of *performance* have been defined as criteria for fitness of application or use, and artifacts instantiating these are called *solutions*.

²⁵ More precisely, the ‘products’ are process *performances*, not the processes themselves. Nevertheless, the point that [performances of] processes are not products in the sense we have been using that term is valid with or without the additional precision.

Accomplishing the kinds of purposes relevant to our discussion here always necessitates successful execution of complex operations comprising many distinct integrated processes; they cannot be immediately and directly achieved by atomic actions. That is, actualizations of these kinds of desiderative states invariably depend upon coordinated actualizations of antecedent states. The effectiveness of such operations is, therefore, an essentially distributive feature of the effectiveness characteristics of their constituent processes and actions. Consequently, achieving satisfactory congruences between the final states actualized by these operations and the desiderative states their correlated purposes stipulate mandates subdivision of their measures of effectiveness and subsequent assignment to their constituent processes and actions. This subdivision and assignment is called *allocation*.

Conditions that impede or block the execution of actions or processes constituting complex operations are called *requirements*—capability voids pertaining to actualizations of *antecedent* states of desiderative states. Instrumental requirements are capability voids respecting *constituent* actions or processes of complex operations—they are entailments of instrumental needs or other instrumental requirements and thus are exigencies of artifact *elements*. Fulfillment of requirements is *synthesis*—an outcome of effective artifact *decomposition*, *element* realization, and *integration*. The performance characteristics of such artifacts are, therefore, functionally dependent upon the performance characteristics and integrity of their elements and are aggregates of these. Consequently, achieving requisite levels of instrumental performance necessitates systematic allocations of those measures to artifact elements.

Needs, requirements, operational measures of effectiveness, functional performance measures, and the multi-dimensional networks of dependence, allocation, and containment relationships among these collectively constitute *functional* and *realization architectures* of product or process systems. These architectures are the *ultimate determinants* of all structural configurations and systemic characteristics. They are the rationale for *why* artifacts are they way they are; they are the ultimate analytic factors to which all synthetic facts are traceable; they are their genesis and their final reasons. Nevertheless, existing product representation systems completely fail to represent them, as these architectures and their elements are altogether beyond their representational capabilities. And, nothing demonstrates the gravity of this unfulfilled representational need more effectively than the pervasive and persistent problems with the engineering change process—especially its impact analysis and propagation subprocesses.

3.2 Product Representation Requirements

We have divided the requirements antecedent to the product representation needs outlined in paragraph 3.1 into three distinct groups—*metastructures*, functional *capabilities*, and superordinate *enterprise operating system* requirements. The first two groups are presented below in paragraphs 3.2.1 through 3.2.6; the latter are partially sketched in Section 4, Results Part II.

We limited our analysis to those requirements which are critical, formal, and immediately antecedent to the identified needs, thereby excluding several classes of operational requirements from our analysis. Specifically, we did not derive or develop characterizations of performance, utilization, and life-cycle “illity” requirements that would naturally follow from an empirical analysis, focused on specifying sufficiency conditions for synthesis and implementation. Again, our aim was to fill a requirements definition void by identifying core *formal* requirements for *all possible* solutions to these particular product representation needs, independently of any requisite technical, economic, or strategic considerations. Nevertheless, a complete and effective product representation solution will have to satisfy *all* requirements deriving from these and other related needs, not just those presented here.²⁶ However, any viable information system element of such a solution will necessarily satisfy the first two groups of requirements, and any total solution will necessarily satisfy all three.

3.2.1 Metastructure

The 17 specific relation types depicted in Table 1 below collectively constitute the necessary informative metastructures required to satisfy the product representation needs previously outlined in paragraph 3.1. That is, any information system solution to those needs must explicitly implement representations of these 17 structures and provide facilities for creating, modifying and querying instances of them.

The metastructures are divided into two distinct groups. *Core* metastructures comprise the essential elements required to represent the basic mereological structure and morphology of any entity at all. *Adjunct* metastructures encompass three subsidiary groups. *Intrinsic* adjuncts constitute those required to explicitly represent systematic variability among the members of a given entity type. *Extrinsic* adjunct metastructures constitute those required to explicitly represent relations between elements of one structural configuration and its counterparts in others, and *intertypic* adjuncts constitute those required to explicitly represent relationships between the taxonomically distinct entity types in a complete product representation.

We do not address schematic or metaschematic product representation content requirements over and above those pertaining to the specific structures in Table 1. That is, we are not concerned with, nor do we specify, any entity types or attributes of entity types excepting those which are directly entailed by the definitions of the above relation types. This is not to say that such characteristics are unimportant to a product representation system: they are very important indeed. However, our focus here is on previously unidentified or incompletely articulated requirements deriving from the needs identified above, not on recapitulating those which have already been identified by others.²⁷

²⁶ Product representation *visibility* requirements deriving from certain stakeholder-specific needs are an illustrative example. Customers and Suppliers (and in some cases, certain regulatory agencies), need access to product representations, albeit for different purposes. Many so-called ‘customer relationship management’ and ‘supply chain integration’ problems are symptomatic of the narrow focus and limited capabilities of existing product data management systems to satisfy these requirements, especially those which are unique to complex manufacturing environments such as aerospace and defense. Both end-user visualization and programmatic access requirements also fall under this group; they are actually agency type-specific product representation visibility requirements as well. These were deliberately excluded from our analysis, as they are neither formal nor directly antecedent of the identified needs. Nevertheless, they do constitute a necessary set of satisfaction conditions for a complete and effective product representation solution.

²⁷ For example, a massive amount of work to define the entity type `PRODUCT DEFINITION`, its attributes, variants, and a host of related entity types, has already been produced by efforts such as `STEP`, as we mentioned before. Another example is the entity type `DOCUMENT`, a critical relatum of several relation types we define here. The `SGML DocBook DTD` (Document Type Definition) [23], is a monumental representation of a ‘tech pub’ variant of that entity type. Our objective is to fill a requirements definition void, not to restate the results of these and many other similar efforts.

	TYPE	RELATION OF	RELATION BETWEEN↔	EXAMPLES
CORE	1 COMPOSITION	Containment	Entities↔Elements	SYSTEMS↔SUBSYSTEMS; ASSEMBLIES↔COMPONENTS
	2 CONSTITUTION	Materiality	Entities↔Materials	PARTS↔RAW MATERIALS; COMPOUNDS↔CHEM ELMTS
	3 INHERENCE	Characterization	Entities↔Attributes	PARTS↔FEATURES; SYSTEMS↔PERFORMANCE CHARS
	4 QUALIFICATION	Conditionality	Relations↔Antecedents	NEEDS↔REQMTS; PROCS↔INPUTS; COMPTNS↔EFFIES
	5 QUANTIFICATION	Multeity	Relata↔Quantities	TPMS↔TPM VALUES; WHERE-USED QUANTITIES
INTRINSIC	6 EQUIVALENCE	Substitutionality	Relata↔Substituends	MIL-P-18177 TYPE G10↔LP-509 CLASS II GRADE A
	7 ALTERNATION	Optionality	Relata↔Alternatives	F-16 AVIONICS↔COUNTRY-SPECIFIC PACKAGES
	8 VARIATION	Diversity	Baselines↔Variants	JSF↔(AF VARIANT,STOL VARIANT)
	9 ORDER	Positionality	Entities↔Positions	ASSEMBLY ORDER; SERVICE PRECEDENCE
	10 TRANSFORMATION	Development	Entities↔States	C-130H↔C-130J; LINUX KERNEL V2.0.3↔V2.0.4
EXTRINSIC	11 ARTICULATION	Separation	Entities↔Realizations	NOZZLE ASSEMBLY↔(THROAT,CHAMBER,EXIT CONE)
	12 FACTORIZATION	Abstraction		(GEAR ₁ ,GEAR ₂ ,GEAR ₃)↔GEAR BLANK
	13 CONSOLIDATION	Integration		(CHIPS,BOARDS,HOUSING)↔AVIONICS LRU
INTERTPIC	14 INVOLVEMENT	Participation	Continuants↔Occurs	PRODUCTS↔FUNCTIONS; ATTRIBUTES↔TEST PROCS
	15 CAPACITATION	Provisionment	Involvements↔Tools;Agencies	F-18E/F ASSEMBLY↔{FIXTURES; JIGS; ASSEMBLERS}
	16 REPRESENTATION	Objectification	Entities↔Representations	PRDCTS↔{ENG DWGS,DATA}; PROCS↔{TECH MNLS,CODE}
	17 DESIGNATION	Nominalization	Entities↔Designators	PARTS↔PART NUMBERS; AIRCRAFT↔TAIL NUMBERS

Table 1. Product Representation Metastructure Relation Types

Our descriptions of requirements for the core metastructures is extremely detailed and is presented in paragraph 3.2.2. Our presentation of requirements for the adjunct metastructures summarizes their formal definitions as those stood at the end of the MEREOS program. To support our post-MEREOS program enterprise engineering activities, we have subsequently launched an effort under our own auspices to develop new and much more comprehensive characterizations of the adjunct metastructures. We will make the new requirements specifications publicly available when this effort is completed.

Finally, we should stress that our focus is on requirements, not implementation issues such as the choice of one database management system, geometric modeling system, or programming language over another. While some of these may be better suited than others to implementing the formal requirements we define here, our purpose is to articulate *what* these requirements are, not *how* they might be effectively synthesized into an implementation. In a word, our objective is to define the *formal* structures required to address the previously presented informatic needs.

3.2.1.1 SOA Syntax and Notational Conventions

The metastructures in the above table could be rendered in a variety of ways for our formal descriptive purposes. We have chosen to present them using state-of-affairs (soa) syntax and semantics, as these are satisfactory for the task at hand, and the notation is easily assimilated, interpreted and implementationally neutral.

There are two variants of soa—*plural* and *monadic*. A plural soa is a structure of the form:

$$\langle R, \langle x_1, \dots, x_n \rangle \rangle$$

where R designates a *relation type*, $\langle x_1, \dots, x_n \rangle$ designates a *member* of relation type R , and x_i designates some entity standing in the i^{th} place of relation R . A monadic soa is a structure of the form:

$$\langle \phi, \langle x \rangle \rangle$$

where ϕ designates a *property type*, and x designates some entity instantiating that property.²⁸ Examples of the plural and monadic variants are:

$$\begin{aligned} &\langle \text{Loves}, \langle \text{John}, \text{Mary} \rangle \rangle && \text{or} && \langle \text{Intelligent}, \langle \text{Mary} \rangle \rangle; \\ &\langle +, \langle 2, 2, 4 \rangle \rangle && \text{or} && \langle \text{Prime}, \langle 3 \rangle \rangle; \text{ and,} \\ &\langle \text{Composition}, \langle \text{F-22}_{5005}, \text{Engine}_{\text{pw042}} \rangle \rangle. \end{aligned}$$

Note we use the generic term *extant* to designate either the n -tuple or 1 -tuple in soas; thus $\langle \text{F-22}_{5005}, \text{Engine}_{\text{pw042}} \rangle$ in the soa $\langle \text{Composition}, \langle \text{F-22}_{5005}, \text{Engine}_{\text{pw042}} \rangle \rangle$ or $\langle \text{Mary} \rangle$ in the soa $\langle \text{Intelligent}, \langle \text{Mary} \rangle \rangle$ are called the “extants” of those soas. Two or more soas whose extants encompass at least one element in common are called *co-incident*; those whose extants encompass exactly the same elements are called *co-extensive*.

■ SOA Schemes

Since we are using soas to describe product representation metastructures, definitions will be rendered in terms of soa *schemes* rather than soa instances, the latter being used only as examples. An soa scheme is a structure of the form:

$$\langle R, \langle \text{ATTRIBUTE}_1, \dots, \text{ATTRIBUTE}_n \rangle \rangle \quad \text{or} \quad \langle \phi, \langle \text{EXEMPLAR} \rangle \rangle$$

where again R designates a relation type and ATTRIBUTE_i designates an entity *type* whose *instances* stand in the i^{th} place of relation R ; or where ϕ designates a property type, and EXEMPLAR designates some entity *type* whose instances instantiate that property. An example of such a scheme for the COMPOSITION relation is:

$$\langle \text{Composition}, \langle \text{Complex}, \text{Element} \rangle \rangle.$$

SOA schemes as we will be rendering them are really syntactical shortcuts for a much more complex form of soa whose relation is DELIMITATION, and whose extension comprises types necessary to *define* a class of soas, such as the relation type, its attributes, relevant qualification/quantification conditions, possible variants, and context element types. Where applicable these other elements are defined textually rather than in soa syntax.²⁹

We will use the generic term *extension* to designate the schematic counterpart of an soa extant; thus $\langle \text{Complex}, \text{Element} \rangle$ in the scheme $\langle \text{Composition}, \langle \text{Complex}, \text{Element} \rangle \rangle$ or $\langle \text{IntentionalBeing} \rangle$ in the scheme $\langle \text{Intelligent}, \langle \text{Intentional Being} \rangle \rangle$ are called the “extensions” of those soa schemes. Again, this is really a notational shortcut for $\{ \langle x_1, \dots, x_i \rangle_1, \dots, \langle x_k, \dots, x_w \rangle_n \}$ or $\{ \langle x \rangle, \dots, \langle y \rangle, \dots, \langle z \rangle \}$ —that is, the set of all n -tuples or set of all 1 -tuples constituting the extensions of a given type R or ϕ . Two or more soa schemes whose extensions encompass at least one entity type (attribute or exemplar) in common are called *congruent*; those whose extants encompass exactly the same types are called *conspecific*.

²⁸ In LISP speak an soa is a *list*; in relational database speak an soa extant $\langle x_1, \dots, x_n \rangle$ is called a *row* or *tuple*; etc.

²⁹ There are really two reasons why we have employed this DELIMITATION shorthand. First of all, it is much less distracting than a complete presentation would be. Secondly, it is adequate for conveying the essential product representation requirements we are concerned with presenting here. A complete delimitation is a very complex structure indeed—the drawing entitled Architecture Element Elements in Figure 30 in Attachment 2 depicts this structure.

■ Adicity, Order, Reflection, and Self-Application

The extensions of soas and soa schemes can be ‘nested’ in two fundamental and two derivative ways. First, an element of the extant of a plural soa can itself be a polyadic plurality or n -tuple:

$\langle \text{Loves}, \langle \text{Mary}, \langle \text{Frank}, \text{Joe}, \text{Bill}, \text{John}, \text{Bob} \rangle \rangle \rangle$

Second, an element of the extant of either plural or monadic soas can itself be an soa:

$\langle \text{Designation}, \langle \text{“John’s Favorite SOA”}, \langle \text{Loves}, \langle \text{John}, \text{Mary} \rangle \rangle \rangle \rangle$ or

$\langle \text{True}, \langle \langle +, \langle 2, 2, 4 \rangle \rangle \rangle \rangle$

In the above examples the Designation and True soas are 2nd-order; the Loves and + soas 1st-order.

There is a specific variant of higher-order soas called *reflection*. A reflective soa is a 2nd-order or above soa that contains an instance of the *same* soa scheme as an element in its extant. For example

$\langle \text{Designation}, \langle \text{“John’s Favorite Designation SOA”}, \langle \text{Designation}, \langle \text{“John’s Favorite SOA”}, \langle \text{Loves}, \langle \text{John}, \text{Mary} \rangle \rangle \rangle \rangle \rangle \rangle$

is reflective in that the object of the nominal designator in the top-level Designation soa is itself a Designation soa. Finally, there is a variant of reflection called *self-application*. A self-applicative soa is a reflective soa that contains *itself* as an element in its extant.

Any of these forms of nesting can potentially occur in a single soa extant, subject to schematic and empirical constraints.

■ SOA Rank

Instances of soa schemes are classified in three fundamental ways. The first is by their relations. The second is by their order as defined in the prior paragraph. The third is the *metasystematic ranks* of the entities in their extants. The two ranks relevant to our purposes here are TAXON and INDIVIDUAL.³⁰ Consider the following two examples:

$\langle \text{Composition}, \langle \text{F-22}_{5005}, \text{Engine}_{\text{pw042}} \rangle \rangle$ and $\langle \text{Composition}, \langle \text{F-22}, \text{Engine} \rangle \rangle$.

The elements constituting the extant of the first Composition instance above clearly designate a particular individual F-22 and engine, respectively, while those of the second do not—they designate the F-22 and ENGINE product types, or *taxa*. Thus the rank of the first soa is INDIVIDUAL, and the rank of the second is TAXON.

The majority of soa instances of rank TAXON represent elements in the *delimitations* of the entity *types* (i.e., attributes) in their extants, such as F-22, ALLOCATION (the process) or REQUIREMENT SPECIFICATION (the document type). For example, the Composition instance of TAXON rank above is defining a mereological containment relation between members of the product taxon F-22 and members of the ENGINE taxon. Soas such as these are the formal analogs of structures implemented by E-BOMS, M-BOMS, and L-BOMS, which represent the structures of product *types* as opposed to those of particular individual products. Again, these soas are syntactical shortcuts for $\langle \text{Delimitation}, \langle \dots \rangle \rangle$ soas—in these cases they are elements of delimitations of entity rather than relation taxa.

Soa instances of rank INDIVIDUAL represent elements of particular members of entity types. For example, the Composition instance of INDIVIDUAL rank above is defining a mereological containment relation between F-22 tail #5005 and Pratt & Whitney engine serial #042. These are the formal analogs of structures implemented by serialized “as-built” and “as-maintained” BOMS, which represent particular structures of particular *instances* of product types.

In some cases elements in the extants of soas can differ in metasystematic rank, making ascription of it to the containing soa less obvious than cases where the rank of all extant elements are the same. For example, the ranks of

$\langle \text{Designation}, \langle \text{“5005”}, \text{theFirstFlightF-22} \rangle \rangle$ and $\langle \text{Designation}, \langle \text{LMF-22Designator}, \text{F-22} \rangle \rangle$

³⁰ There are *four* metasystematic ranks or categories in our own formal system: CATEGORY, TAXON, INDIVIDUAL, and FACT. Thus in principle the relata in any soa instance or scheme could be of *any one* of these four ranks. However in practice this categorial scope is not necessary for multiple BOM reconciliation. We have not, accordingly, imposed it as a product representation requirement here.

are clearly INDIVIDUAL and TAXON, respectively; while the rank of

$\langle \text{Designation}, \langle \text{"Raptor"}, \text{F-22} \rangle \rangle$

is not obvious, as "Raptor" is a particular designator, but F-22 is clearly a product taxon, not a particular airplane. We will address the question of rank of such 'mixed rank' soas on a case-by-case basis where it is important to specifying requirements.

■ Modal SOAs and Contexts

Any 2nd-order or above soa modifies or *modalizes* the soa or soas in its instance, and the plurality of modal soas immediately containing a given soa as elements in their extants collectively constitute the *context* of that soa. Rendering this as an explicit soa itself, a context is a structure of the form

$\langle \text{Context}, \langle \text{ObjectSOA}, \langle \text{ModalSOA}_1, \dots, \text{ModalSOA}_n \rangle \rangle \rangle$

where Context is a special form, where ObjectSOA is the target soa modalized by one or more n^{th} -order soas, and where ModalSOA_i is an soa containing ObjectSOA as an element of its extant. An explicitly contextualized soa is rendered using the form:

$$\begin{array}{l} \langle [], \langle R, \langle x_1, \dots, x_n \rangle \rangle \rangle \\ \left| \begin{array}{l} \langle R^w, \langle x_1, \dots, \text{SOA}_x, \dots, x_n \rangle \rangle \\ \langle \phi^a, \langle \text{SOA}_x \rangle \rangle \end{array} \right. \end{array}$$

where [] designates the context of the target soa, $\langle R^w, \langle x_1, \dots, \text{SOA}_x, \dots, x_n \rangle \rangle$ and $\langle \phi^a, \langle \text{SOA}_x \rangle \rangle$ designate the $[n+1]^{\text{th}}$ -order modalizing soas, and where SOA_x designates the n^{th} -order soa.

We will use the term *intension* to designate the schematic counterpart of an soa context.

■ Complementarity

By default all soas are positive; that is, they represent affirmations of facts. Thus the soa

$\langle \text{Composition}, \langle \text{F-22}_{5005}, \text{Engine}_{\text{pw042}} \rangle \rangle$

represents a containment relation between a particular F-22 and a particular Pratt & Whitney engine, asserting that engine serial number pw042 *is*, in fact, a component element of F-22 tail number 5005.

However, it is sometimes necessary to *explicitly* represent a *negative* fact—that is, a fact of *opposition*. The special modal form called COMPLEMENTARITY—the modality of *polarity*—together with its two variants THETIC (positive), denoted by the symbol \wp , and KENONIC (negative), denoted by the symbol \wp^* , is employed for this purpose. Thus the explicitly designated KENONIC soa

$$\begin{array}{l} \langle [], \langle \text{Composition}, \langle \text{F-22}_{5005}, \text{Engine}_{\text{pw042}} \rangle \rangle \rangle \\ \left| \begin{array}{l} \langle \wp^*, \langle \text{SOA}_{\text{pw}} \rangle \rangle \\ \langle \dots \end{array} \right. \end{array}$$

represents the fact that engine serial number pw042 *is not* a component element of F-22 tail number 5005.

Note that KENONIC and uniformly rank INDIVIDUAL soas represent particular negations, and that KENONIC and uniformly rank TAXON soas represent specific delimitative exclusions.

■ Identities

As we mentioned in our introductory remarks, the informatic metastructures required to satisfy the product representation needs we are concerned with here are all relations between entities, and we are accordingly only concerned with defining requirements pertaining to those and their immediately entailed content requirements. Some of these involve related entity types and their attributes, and not all entities are relations or properties. For example, neither the product type F-22 nor the instance of that type F-22 tail #5005 are relations between entities or properties of them. While we are not concerned with enumerating the characteristics of the F-22 or indeed PRODUCT itself, we will

on occasion need to specify requirements for entities which are not relations or properties. Such entities are rendered by instances of the *IDENTITY* SOA scheme:

```

⟨[ ],⟨Identity,⟨⟩⟩⟩
|
|⟨Designation,⟨“x”,theIdentitySOA⟩⟩
|⟨Rm,⟨f,...,theIdentitySOA,...g⟩⟩
|⟨φa,⟨theIdentitySOA⟩⟩
|⟨Rw,⟨s,...,theIdentitySOA,...q⟩⟩
|etc.

```

where *Identity* is an extensionless special form. Note that soas containing *x* as elements in their extants actually contain the soa *theIdentitySOA*—that is, “*x*” in our standard soa notation is really a syntactical shortcut for an *IDENTITY* SOA instance, modalized by a *DESIGNATION* SOA with “*x*” as its designator.³¹

■ Particulars

The *IDENTITY* special form encompasses four specific variants, called “particulars,” derived by varying *COMPLEMENTARITY*, metasystematic rank, and adicity.

1. BARE particular – the THETIC (ϑ) rank INDIVIDUAL variant of *IDENTITY*. Instances of this variant represent pure extensionless facts of positive individuality, connoted by the terms “this” or “these.”
2. ARBITRARY particular – the THETIC (ϑ) rank TAXON variant of *IDENTITY*. Instances of this variant represent pure extensionless facts of positive taxicity, connoted by the terms “anything” or “any things.”
3. NULL particular – the KENONIC (ϑ) rank INDIVIDUAL variant of *IDENTITY*. Instances of this variant represent pure extensionless facts of negation, connoted by the terms “nothing” or “no things.”
4. INDETERMINATE particular – the KENONIC (ϑ) rank TAXON variant of *IDENTITY*. Instances of this variant represent pure extensionless facts of exclusion, connoted by the terms “something” or “some things.”

■ Indexicals

It is sometimes necessary for an element in the extant of one soa to explicitly address or ‘point at’ or ‘virtually contain’ one or more *elements* of another soa, *in situ*. The *INDEXICALIZATION* special form³² is used to accomplish this, and it is a structure of the general form:

SourceSoa_{*x*}: ⟨[],⟨R,⟨x₁,...,⟨Indexicalization,⟨TargetSoa,ReferentCoordinate⟩⟩,...,x_{*n*}⟩⟩

where *SourceSoa_x* designates the soa containing the indexical element, *TargetSoa* designates the soa whose element is being situationally referred to as an extant element in *SourceSoa_x*, and where *ReferentCoordinate* designates a target soa-specific coordinate, and sometimes an soa instance extant-specific configuration coordinate. This coordinate either designates the ordinal position of the referent element in the target soa instance, or is a unique or uniquely resolvable nominal designation of that element in that position. Note that all indexicals are necessarily embedded; that is, they are necessarily extant elements of other soas. They are at least 2nd-order, thus they are also context elements of their target soas.

³¹ From this we can state a refinement to our prior definitions of 1st-order and 2nd-order soas. A 1st-order soa is one whose extant elements consist solely of *IDENTITY* soas or “terminal values;” nominal designators being examples of these. A 2nd-order or above soa is one containing at least one 1st-order or above soa as an extant element.

³² Strictly speaking *INDEXICALIZATION* is a variant of the *DESIGNATION* relation type defined in paragraph 3.2.5.4. We have defined it here as it will be used extensively prior to that paragraph.

I Graphics

Representation system metastructures are intrinsically abstract, intricately inter-related, and correspondingly difficult to communicate. Over the course of the MEREOS program, we developed a series of graphics to assist in developing our conceptualizations of these structures. Although there is no substitute for unambiguous syntax and illustrative examples, we have found these visualizations helpful and thus have incorporated them into the definitions presented here.³³

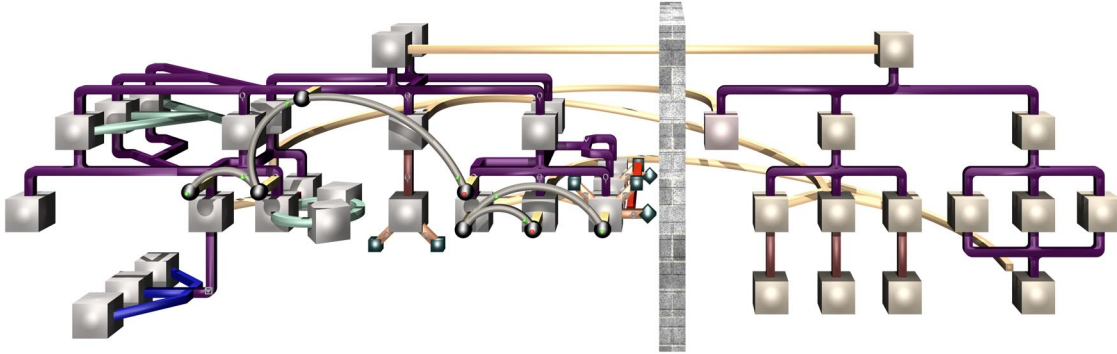


Figure 3. Metastructures in Graphical Form

³³ A complete and much larger version of Figure 3 above is included in Figure 33 in Attachment 2.

I Symbology

For the sake of typographical brevity we will, on occasion, use certain symbols in conjunction with our SOA notation, as follows:

\coloneqq	assignment	- attribute type delimitation or attribute instance entity identification
\equiv	identity	- logical equivalence; also EQUIVALENCE relation as defined in 3.2.3.1 below
\neq	difference	- logical inequivalence
\neg	negation	- logical NOT
\wedge	conjunction	- logical AND
\vee	disjunction	- logical OR
∇	exclusion	- logical XOR
\diamond	possible	- an actual inoperative fact
\diamond	impossible	- an <i>unactual</i> inoperative fact
\diamond	contingent	- a conditionally operant fact
\square	necessary	- a fact nomologically related to the operance of one or more facts
\square	extrinsic	- a fact nomologically <i>unrelated</i> to the operance of one or more facts
\wp	thetic	- a positive fact
\wp	kenonic	- a negative fact
\bullet	bare fact	- positive IDENTITY SOA of rank INDIVIDUAL
\blacksquare	arbitrary fact	- positive IDENTITY SOA of rank TAXON
\circ	null fact	- negative IDENTITY SOA of rank INDIVIDUAL
\otimes	indeterminate	- negative IDENTITY SOA of rank TAXON
[]	optionality	- SOA extension configuration or element variation ³⁴
\rightarrow	indexicalization	- as defined on page 31 above
λ	composition	- the relation defined in 3.2.2.1 below
\perp	constitution	- the relation defined in 3.2.2.2 below
\odot	inherence	- the relation defined in 3.2.2.3 below
\supset	antecedence	- variant of the QUALIFICATION relation defined in 3.2.2.4 below
$\overline{\supset}$	consequence	- variant of the QUALIFICATION relation defined in 3.2.2.4 below
#	quantification	- the relation defined in 3.2.2.5 below
[]	involvement	- the relation defined in 3.2.5.1 below

Uppercase letters (A,B,C,...Z) will unless otherwise indicated be used as variables designating entities of metasystematic rank TAXON. Lowercase letters (a,b,c,...,z) will unless otherwise indicated be used as variables designating entities of metasystematic rank INDIVIDUAL.

³⁴ Square brackets within SOA extension definitions— $\langle R_x, \langle \text{Attribute}_1, [\text{Attribute}_2 := \text{Variant}_p \nabla \text{Variant}_q], \dots \rangle \rangle$ —are used to indicate variability among extants of instances of a relation type and to define the variations; that is, to mark the fact that extensional variants or ‘subtypes’ of that relation type exist. Hence the expression $[\text{Attribute}_2 := \text{Attribute}_p \nabla \text{Attribute}_q]$ designates the fact that relation type R_x comprises two extensional subtypes, as entities standing in the Attribute_2 position can be instances of Variant_p XOR instances of Variant_q . The ORDER relation type defined in 3.2.3.4 below, which encompasses four distinct extensional variants, is an illustrative example.

3.2.1.2 Schematic Metastructure Requirements

Several requirements pertaining to particular metastructures are actually metatype-specific configurations of more general requirements that apply to some or all of the metastructures, albeit in different ways and to differing degrees. Schematic characterizations of these requirements are presented here.

■ Polymorphism

Unless otherwise specified, any instance of any metastructure type can contain an instance of any other metatype as an extant element. This is called *contingent polymorphism*. Delimitations of some metastructure types or their variants entail that their instances necessarily contain instances of other metatypes. This is called *necessary polymorphism*.

Instances of any of the 17 metatypes whose extants contain an INDEXICALIZATION element are examples satisfying the contingent polymorphism requirement scheme. The QUALIFICATION and QUANTIFICATION metatypes (defined in paragraphs 3.2.2.4 and 3.2.2.5) are examples of metatypes satisfying the necessary polymorphism requirement scheme.

■ Reflection

Unless otherwise specified, any instance of any metastructure type can contain an instance of that same metatype as an extant element. This is called *contingent reflection*, and its realization in terms of soa structures was presented on page 29 above. Delimitations of some metastructure types or their variants entail that their instances necessarily contain instances of the same metatypes. This is called *necessary reflection*.

The non-terminal variants of the ORDER metatype (defined in paragraph 3.2.3.4) are examples of metatypes satisfying the necessary reflection requirement scheme.

■ Self-Application

An instance of a metastructure type may contain *itself* as an element of its extant if and *only if* explicitly specified. This is called *self-application*. Its realization in terms of soa structures was also presented on page 29 above.

■ Supplemantarity

The instances of some metastructure types or variants are existentially exiguous. That is, the existence of a single particular instance of these types necessarily entails the existence of at least one other *co-incident* instance of that same type. This is called *extensive supplemantarity*. A stronger variant of this necessarily entails the existence of at least one other co-incident *but detached* instance of that same type. This is called *disjoint supplemantarity*.

Consider some metatype R and an instance of it $\langle R, \langle x, y \rangle \rangle$. If R is extensively supplemantational, then necessarily there exists at least one other instance:

$$\langle R, \langle x, z \rangle \rangle \vee \langle R, \langle z, x \rangle \rangle.$$

If R is disjointly supplemantational, and an instance of it $\langle R, \langle x, y \rangle \rangle$ exists, then necessarily there exists at least one other instance:

$$\langle R, \langle x, z \rangle \rangle: \text{such that } \neg \langle R, \langle y, z \rangle \rangle \text{ and } \neg \langle R, \langle z, y \rangle \rangle \vee \langle R, \langle z, x \rangle \rangle: \text{such that } \neg \langle R, \langle y, z \rangle \rangle \text{ and } \neg \langle R, \langle z, y \rangle \rangle.^{35}$$

Rank TAXON instances of the COMPOSITION metatype are examples satisfying the extensive supplemantarity requirement scheme. Rank INDIVIDUAL instances of this metatype are examples satisfying the disjoint supplemantarity requirement scheme.

The instances of some attributes delimiting extensions of some metastructure types are necessarily minimally dyadic. This is called *positional supplemantarity*. Consider some metatype R and an attribute of its extension Q, such that $\langle R, \langle A, \dots, Q, \dots, S \rangle \rangle$. If Q is positionally supplemantational, then every instance of Q in the extant of every instance of R is necessarily at least a 2-tuple:

³⁵ This is a generalization and informal articulation in soa syntax of the formally rendered Weak Supplementation Principle (wsp) in Simons [20.2].

$$\langle R, \langle x_i, \langle q_1, q_2, \dots, q_n \rangle, s_k \rangle \rangle$$

such that q_1, q_2, \dots, q_n all designate distinct entities. The SUBSTITUEND attribute of the EQUIVALENCE metatype (defined in paragraph 3.2.3.1) is one type satisfying the positional supplementarity requirement scheme.

■ Rank Coordination

All elements of all metatype instances are either of TAXON or INDIVIDUAL metasystematic rank. However as described on page 29 above, in general one extant element in a given soa can be of a different rank than another element within that soa. The extant elements of instances of most metastructure types must be the same metasystematic rank. For 1st-order instances this is called *necessary rank uniformity*; for types encompassing higher-order instances (polymorphic, reflective, or both), this is called *complete rank uniformity*.

For example, consider the 2nd-order polymorphic composite structure represented by the soa below.

$$\langle \text{Composition}, \langle \langle \text{Constitution}, \langle A, B \rangle \rangle, \langle \text{Constitution}, \langle W, Z \rangle \rangle \rangle \rangle$$

This soa represents a containment relation between two dependence relations, signifying that the material dependence of W on Z—i.e., $\langle \text{Constitution}, \langle W, Z \rangle \rangle$ —is a component element of the material dependence of A on B—i.e., $\langle \text{Constitution}, \langle A, B \rangle \rangle$. Complete rank uniformity stipulates that *either* A, B, W, and Z are *all* of rank TAXON *or* are all of rank INDIVIDUAL. The VARIATION metatype (defined in paragraph 3.2.3.3) is one example satisfying this requirement scheme.

■ Cyclicity

A given set of soa instances delineated by some relation R can be treated as a directed graph. One graph-theoretic property and its converse—cyclicity and acyclicity—are invoked in some metastructure requirements, so we define them here. An apparent form of cyclicity, which is an artifact of our use of soa schemes and rank TAXON instances as DELIMITATION shortcuts is also defined.

CYCLOCITY

The extants of the members of any set of plural soas delimited by a given relation R can be rendered as a single sequence of unique extant elements:

$$\{ \langle R, \langle x_1, x_2 \rangle \rangle, \langle R, \langle x_2, x_3 \rangle \rangle, \dots, \langle R, \langle x_{n-1}, x_n \rangle \rangle \} \rightarrow \langle x_1, x_2, x_3, \dots, x_{n-1}, x_n \rangle$$

Any such sequence is *cyclic* under relation R if and only if:

$$\langle R, \langle x_1, x_2 \rangle \rangle \text{ and } \langle R, \langle x_2, x_3 \rangle \rangle \text{ and } \dots \text{ and } \langle R, \langle x_n, x_1 \rangle \rangle.$$

ACYCLOCITY

Any such sequence is *acyclic* if and only if it is *not* cyclic under a given relation R.

DELIMITATIVE CYCLOCITY

Any set of uniformly rank TAXON soa instances $\{ \langle R, \langle T_1, T_2 \rangle \rangle, \langle R, \langle T_2, T_3 \rangle \rangle, \dots, \langle R, \langle T_n, T_1 \rangle \rangle \}$ is cyclic given the above definition. However, as we stated in our presentation of soa syntax and notational conventions above, the majority of soa instances of rank TAXON represent facts of *delimitation* rather than facts of specific relatedness, and thus any such cycle is ambiguous. That is, it could be interpreted to represent that a member x of taxon T_i can stand indirectly in relation R to itself—i.e., that instances of that metatype *can* be cyclic. On the other hand, it could also be interpreted to represent that *some* member x of taxon T_i can stand in relation R to *another* member y of taxon T_i , but that x *cannot* stand in that relation to itself. This latter case, called *delimitative cyclicity*, expresses a *taxonomic generalization* concerning relatedness of members of taxon T_i , in respect to relation R.

In many cases constraints apply on members of T_i with respect to standing in some attribute position of relation R. This is called *conditional rank taxon cyclicity*. In some cases there are none. This is called *unconditional rank taxon cyclicity*.

■ Existential Status

Unless otherwise specified, the COMPLEMENTARITY modality of any instance of any metastructure type can be either THETIC (\wp) or KENONIC (\wp), and via INDEXICALIZATION instances, so can any extant element of any instance. However, as noted above, all instances and extant elements are THETIC unless their COMPLEMENTARITY modalities are explicitly designated as KENONIC (\wp). Thus any SOA W

$$SOA_W: \langle R, \langle x, y \rangle \rangle$$

lacking an explicit representation of COMPLEMENTARITY modality shall be interpreted as synonymous with an explicitly modalized SOA

$$\begin{array}{l} \langle [], \langle R, \langle x_1, \dots, x_n \rangle \rangle \rangle \\ \left| \right. \\ \langle \wp, \langle SOA_W \rangle \rangle \\ \left| \right. \\ \langle \dots \end{array}$$

and any INDEXICALIZATION instance designating an element in an SOA_W

$$\langle R, \langle z, \langle \text{Indexicalization}, \langle SOA_W, "y" \rangle \rangle \rangle$$

without itself having an explicit representation of COMPLEMENTARITY modality shall be interpreted as synonymous with an explicitly modalized INDEXICALIZATION instance

$$\begin{array}{l} \langle R, \langle z, \langle [], \langle \text{Indexicalization}, \langle SOA_W, "y" \rangle \rangle \rangle \rangle \\ \left| \right. \\ \langle \wp, \langle \text{INDEXICALIZATIONSOA} \rangle \rangle \\ \left| \right. \\ \langle \dots \end{array}$$

Any element of any instance of any metastructure type can, unless otherwise specified, be either an ARBITRARY or an INDETERMINATE particular. Thus in accordance with the definitions of these IDENTITY variants above, an SOA instance of the form

$$\langle R, \langle x, \bullet \rangle \rangle$$

shall be interpreted as a fact that entity x stands in relation R to *any* [arbitrary] entity capable of standing in the indicated monadic attribute position, and an SOA instance of the form

$$\langle R, \langle x, \wp \rangle \rangle$$

shall be interpreted as a fact that entity x stands in relation R to *some* [indeterminate] entity capable of standing in the indicated monadic attribute position, and finally that an SOA instance of the form

$$\langle R, \langle x, \langle \bullet \rangle \rangle \rangle \text{ or } \langle R, \langle x, \langle \wp \rangle \rangle \rangle$$

shall be interpreted as a fact that entity x stands in relation R either to any or some entities collectively capable of standing in the indicated polyadic attribute position, respectively.

An extant element of an instance of a metastructure type can be a NULL particular (\circ) *if and only if* explicitly specified.

3.2.2 Core Metastructures

	TYPE	RELATION OF	RELATION BETWEEN↔	EXAMPLES
CORE	1 COMPOSITION	<i>Containment</i>	Entities↔Elements	SYSTEMS↔SUBSYSTEMS; ASSEMBLIES↔COMPONENTS
	2 CONSTITUTION	<i>Materiality</i>	Entities↔Materials	PARTS↔RAW MATERIALS; COMPOUNDS↔CHEM ELMTS
	3 INHERENCE	<i>Characterization</i>	Entities↔Attributes	PARTS↔FEATURES; SYSTEMS↔PERFORMANCE CHARS
	4 QUALIFICATION	<i>Conditionality</i>	Relations↔Antecedents	NEEDS↔REQMTS; PROCS↔INPUTS; COMPSTNS↔EFFIES
	5 QUANTIFICATION	<i>Multeity</i>	Relata↔Quantities	TPMS↔TPM VALUES; WHERE-USED QUANTITIES

Table 2. Core Relations

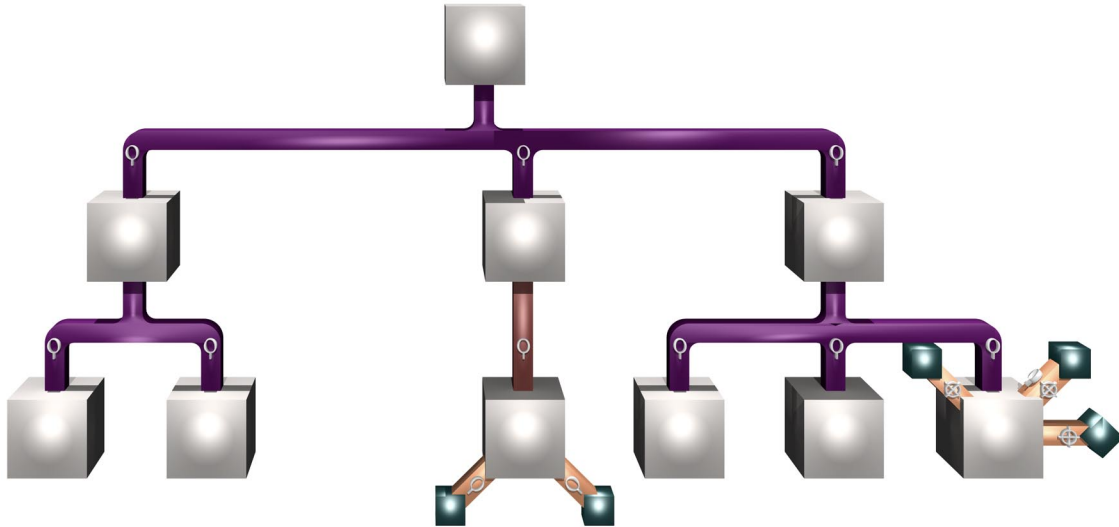


Figure 4. Core Relations

All structures of all artifacts minimally consist in five fundamental types of relations.

1. COMPOSITION —the relation of *mereological containment*, depicted by purple lines in above figure;
2. CONSTITUTION —the relation of *material dependence*, (brown lines, center);
3. INHERENCE —the relation of *exemplification* (orange lines, right side);
4. QUALIFICATION —the relation of *conditionality* (Greek letter ϕ); and,
5. QUANTIFICATION—the relation of *multeity* (not shown).

Any minimally correspondent and capable product representation system must accordingly implement explicit representations of these five relation types and provide facilities for instantiating, modifying, and querying them.

3.2.2.1 COMPOSITION

Composition relations, depicted by the purple lines in Figure 4 above, are structures of the form:

$\langle \text{Composition}, \langle \text{Complex}, [\text{Element} := \text{Element} \vee \circ] \rangle \rangle$

where *Composition* designates the Mereological Containment taxon; *Complex* designates entities that contain component elements; where *Element* designates entities that are component elements; and where \circ designates an instance of the NULL PARTICULAR variant of *IDENTITY*.

Mereological Containment is a fundamental delimitative element of structure. *COMPOSITION* is, accordingly, a critical metatype required for representing it. This relation is one of the two primary types represented by classical BOMS—the other being *CONSTITUTION*, defined in paragraph 3.2.2.2 below. Examples of *COMPOSITION* relations are:

$\langle \lambda, \langle \text{Assembly}_x, \text{Component}_y \rangle \rangle$	- a typical <i>CONTAINMENT</i> relation in BOMS
$\langle \lambda, \langle \text{Document}_w, \text{Paragraph}_z \rangle \rangle$	- e.g., this document and paragraph 3.3.3
$\langle \lambda, \langle \text{Process}_q, \text{Phase}_r \rangle \rangle$	- e.g., Product Realization and Product Definition

where the symbol “ λ ” denotes *COMPOSITION* as noted on page 33.

■ Composition Variants

There are two infraspecific variants of *Containment*—Integral and Distributive. Facts of the Integral variant are relations between mereologically unified complexes and their components, such as aircraft fuselages, microprocessor chips, polysyllabic words, and uninterruptible multi-phased processes. Facts of the Distributive variant are relations between mereologically dispersed complexes and their components, such as flight control subsystems, brake systems on automobiles, and texts of multi-volume documents. As of this writing we were unable to develop satisfactorily definitive delimitations and identification keys for these variants. We cannot therefore stipulate representation of them and their differences as a requirement here. Nevertheless, the correlate complexes differ in some of their analytic, constructive, and logistical attributes, and once a formal characterization of these is developed, any minimally correspondent and capable product representation system should be enhanced to explicitly represent these variants and to provide facilities for differentiating, instantiating, modifying, and querying them.

Integral variant complexes can be in one of two mereological configurations. These are *Partitioned* and *Bracteal*. Entities standing in the *Complex* attribute position of 1st-order *COMPOSITION* instances containing NULL particulars (“ \circ ”) in their *Element* attribute positions are formalizations of the *Bracteal* configuration rendered as soas. These instances represent the fact that the entities their *Complex* attribute positions are completely partless—i.e., *atomic*.

■ COMPOSITION Requirements

Composition relations are typically quantified, qualified, and otherwise modalized in a variety of

1. Supplementarity

Rank *TAXON* instances of the *COMPOSITION* metatype are *extensively* supplementational, as defined in paragraph 3.2.1.2 above. Rank *INDIVIDUAL* instances of the composition metatype are *disjointly* supplementational as defined in that paragraph.

2. Complete Rank Uniformity

Entities in the *Complex* and *Element* attribute positions in a particular *COMPOSITION* instance must be of the same metasystematic rank, including all elements of all extants of all meta-structure type instances in every higher-order *COMPOSITION* instance.

3. Cyclicity

A given set of rank *INDIVIDUAL* 1st-order *COMPOSITION* instances must be acyclic. That is, a particular complex cannot directly or indirectly contain itself as a component element.

All *reflective* COMPOSITION instances must be acyclic. That is, a particular COMPOSITION instance cannot directly or indirectly contain itself as a component element.

A given set of rank TAXON 1st-order COMPOSITION instances may be conditionally cyclic. That is, one member of a complex taxon *can* directly or indirectly contain another member of that same taxon as a component element, subject to requisite qualification condition representation.

A given set of COMPOSITION instances containing at least one *polymorphic* higher-order COMPOSITION instance may be cyclic. That is, an entity in the extant of an instance of a metastructure type contained in the extant of a COMPOSITION instance can contain or be contained by that instance.

4. Existential Status

No entity in the Complex attribute position in a COMPOSITION instance containing a NULL in its Element position can be a complex in any other COMPOSITION instance. That is, an atomic entity is an indivisible matter of fact.

3.2.2.2 CONSTITUTION

Constitution relations, depicted by the brown lines in Figure 4 above, are structures of the form:

$\langle \perp, \langle \text{Superstrate}, [\text{Substrate} := \text{Substrate} \vee \circ] \rangle \rangle$

where Constitution designates the Material Dependence relation taxon; Superstrate designates entities that depend on constituent elements; where Substrate designates entities that are constituent entities; and where \circ designates an instance of the NULL PARTICULAR variant of IDENTITY.

Like Containment, Material Dependence is a fundamental delimitative element of structure, and CONSTITUTION is, therefore, a crucial metatype required for representing it. This type is the second of the two primary types represented by classical BOMS. Examples of CONSTITUTION relations are:

$\langle \perp, \langle \text{PartType}_x, \text{RawMaterialType}_y \rangle \rangle$	- a classical MATERIALITY relation in BOMS
$\langle \perp, \langle \text{CompositeMatType}_w, \text{MatrixMatType}_z \rangle \rangle$	- e.g., graphite fiber and epoxy
$\langle \perp, \langle \text{CompoundType}_q, \text{ElementType}_r \rangle \rangle$	- e.g., Water (H ₂ O) and hydrogen

where the symbol " \perp " denotes constitution as noted on page 33.

■ CONSTITUTION Variants

Superstrata can be in one of two substantival configurations. These are Heteronomous and Independent. Entities standing in the Superstrate attribute position of 1st-order CONSTITUTION instances containing NULL particulars (" \circ ") in their Substrate attribute positions are formalizations of the Independent configuration rendered as soas. These instances represent the fact that the entities their Superstrate attribute positions are completely self-subsistent—i.e., *autonomous*.

■ CONSTITUTION Requirements

1. Self-Application

A particular CONSTITUTION instance may contain itself as an element in the Substrate attribute position of its extant. That is, a particular fact of material dependence may constitute itself.

2. Supplemantarity

All self-applicative rank TAXON CONSTITUTION instances are extensively supplementational, as defined in paragraph 3.2.1.2 above. All self-applicative rank INDIVIDUAL CONSTITUTION instances are disjointly supplementational as defined in that paragraph. That is, a particular fact of material dependence may not *completely* constitute itself.

3. Existential Status

No entity in the *Superstrate* attribute position of a *NULL* substrate *CONSTITUTION* instance can be a superstrate in any other *CONSTITUTION* instance. That is, an autonomous entity is an ultimate matter of absolute fact.

3.2.2.3 INHERENCE

Inherence relations, depicted by the orange lines in Figure 4 above, are structures of the form:

$\langle \text{Inherence}, \langle \text{Bearer}, \text{Character}, [\text{Basis} := \text{Basis} \vee \circ] \rangle \rangle$

where *Inherence* designates Exemplification relation taxon; *Bearer* designates entities that are examples of characteristics; where *Character* designates exemplified entities; where *Basis* designates entities that are determinants of characters as exemplified by entities; and where \circ designates an instance of the *NULL PARTICULAR* variant of *IDENTITY*.

Exemplification is the third fundamental delimitative element of structure, and *INHERENCE* is, therefore, a critical metatype required for representing it. Examples of *INHERENCE* relations are:

$\langle \text{“} \langle \text{PartType}_x, \text{ThroughHole}_y, \text{SurfaceOfRevolution}_a \rangle \text{”} \rangle$	- a paradigm geometric feature
$\langle \text{“} \langle \text{Object}_w, \text{Color}_y, \text{Optico-PerceptualStruct}_d \rangle \text{”} \rangle$	- a paradigm ‘secondary quality’
$\langle \text{“} \langle \text{7075-T7}, \text{TensileStrength}, \text{Microstructure}_\lambda \rangle \text{”} \rangle$	- a material property
$\langle \text{“} \langle \text{Process}_z, \text{Reliability}[\text{.98}], \text{ImplementationConfig}_\sigma \rangle \text{”} \rangle$	- a process effectiveness metric
$\langle \text{“} \langle \text{results1.fm}, \text{“} \text{3 INHERENCE ”}, \text{“/art/coreMetaTable.pct”} \rangle \text{”} \rangle$	- i.e., Fig 4 artwork, imported by reference

where the symbol “ “ ” denotes *CONSTITUTION* as noted on page 33.

Delineating features of entities such as topological and geometrical form, physical properties, and other derivative traits, *INHERENCE* is a formalization of structures implemented in geometric modeling, structural analysis, and imaging systems, and is *not* represented by classical BOMs. Nevertheless, representing this relation and its related entity types as integral elements of structure is directly entailed by the amplitude needs described in paragraphs 3.1.1 and 3.1.2 above. That is, particular divergences between multiple genitive configurations of artifacts are grounded in specific artifact characteristics, as we implicitly demonstrated in our discussions of articulation, factoring, and consolidation. Hence representing these characteristics and the relations they stand in is a prerequisite of representing inter-configuration differences as relations between configurations.³⁶ Moreover, artifacts and their elements, including their specific characteristics, are all solutions to instrumental requirements, as we pointed out in our discussion of the requirements representation need on page 24. Thus representing *all* the elements of artifact structure—including specific artifact characteristics—is a prerequisite of representing the physical synthesis of needs and requirements as relations between these and the elements and characters of artifacts that fulfill them.

■ INHERENCE Variants

There are two infraspecific variants of Exemplification—Faceted and Dispositional. Facts of the Faceted variant are relations between entities and spatial features, such as surface deformations and protrusions, holes, faces, and other varieties of topological and geometrical form. Facts of the Dispositional variant are relations between entities and properties, such as material, functional, economic, and psychological attributes—indeed any variety of determinable magnitude. Facts of the Dispositional variant are represented by *INHERENCE* instances with *QUANTIFICATION* instances³⁷ in their *Character* attribute positions. That is, dispositional exemplifications are externalized manifestations of *quantities*, in the most general sense of that term. Facts of the Faceted variant are represented by those with instances of any *other* metatype in their *Character* attribute positions—particularly those with *IDENTITY* instances as characteristics—these being explicit objectifications of

³⁶ Our formalizations of these relations and the roles of *INHERENCE* in those are presented in paragraphs 3.2.4.1, 3.2.4.2, and 3.2.4.3 below.

³⁷ *QUANTIFICATION* is defined in paragraph 3.2.2.5 below.

features qua entities.

Entities exemplified by facts of either Exemplification variant can be in one of the configurations. These are Attributive and Intrinsic. Entities standing in the **Character** attribute position of **INHERENCE** instances containing any instances other than **NULL** particulars (“o”) in their **Basis** attribute positions are formalizations of the Attributive configuration rendered as soas. The first two examples above (i.e., PartType_x and Object_w) are representative of this configuration and the type of characteristic that distinguishes it. These characteristics are *ascriptive*—they are externalized manifestations of entities in **Basis** attribute positions by entities in **Bearer** attribute positions. Entities standing in the **Character** attribute position of **INHERENCE** instances containing **NULL** particulars (“o”) in their **Basis** attribute positions are formalizations of the Intrinsic configuration and the type of characteristic that marks it. These characteristics are primitives—either within the scope of a given representational context or system, or in the actual world. The properties of geometric primitives in modeling systems are examples of the former; the speed of light in a vacuum c is an example of the latter.

Any minimally correspondent and capable product representation system must implement explicit representations of these Exemplification variants and configurations and provide facilities for differentiating, instantiating, modifying, and querying them.

■ INHERENCE Requirements

1. Complete Rank Uniformity

Entities in the **Bearer** and **Character** attribute positions in a particular **INHERENCE** instance must be of the same metasystematic rank, including all entities in those attribute positions in all extants of all metastructure type instances in every higher-order **INHERENCE** instance.

Entities in the **Basis** attribute position in a particular **INHERENCE** instance with bearers rank **TAXON** entities in its **Bearer** and **Character** attribute positions may be of rank **INDIVIDUAL**.

2. Cyclicity

A given set of rank **INDIVIDUAL** 1st-order **INHERENCE** instances must be acyclic. That is, a particular entity must either be an exemplar, an exemplified characteristic, or a genitive basis of a characteristic as exemplified by a given individual entity.

All *reflective* **INHERENCE** instances must be acyclic. That is, a particular **INHERENCE** instance cannot directly or indirectly contain itself as a component element.

A given set of rank **TAXON** 1st-order **INHERENCE** instances can be conditionally cyclic. That is, two members of the same taxon can stand in the **Bearer**, **Character**, or **Basis** attribute positions of a particular **INHERENCE** instance, subject to requisite qualification condition representation.

A given set of **INHERENCE** instances containing at least one *polymorphic* higher-order **INHERENCE** instance may be cyclic.

3. Existential Status

No entity in the **Character** attribute position in a **INHERENCE** instance containing a **NULL** in its **Basis** position can be a characteristic in any Attributive configuration **INHERENCE** instance. That is, an atomic entity is an indivisible matter of fact.

3.2.2.4 QUALIFICATION

Qualification relations are structures of the form:

$$\langle \text{Qualification}, \langle \text{Correlate}, [\text{Evolute} := \text{Evolute} \vee \circ], [\text{Adject} := \text{Adject} \vee \langle \text{Adject}_1, \dots, \text{Adject}_n \rangle \vee \circ] \rangle \rangle$$

where *Qualification* designates the Conditionality taxon; *Correlate* designates either contingently operant entities or determinants of contingently operant entities; where *Evolute* designates TRANSFORMATION instances representing ontogenetic elements of correlate entities as defined in paragraph 3.2.3.5 below; where *Adject_i* designates an entity that is nomologically related to the operance status of its correlate entity; and where *Null* designates an instance of the NULL PARTICULAR variant of IDENTITY. This relation type is a formalization of “causal dependence” or *contingency* and therefore is, strictly speaking, a co-variant of CONSTITUTION. Its variants, configurations, and relata—one of these latter being a formalization of “effectivity”—are essential for representing provisionally or periodically operant matters of fact and are, accordingly, cornerstones of representational correspondence.

■ QUALIFICATION Variants

There are two positional variants of Conditionality—Subvenient and Supervenient. There are two distinct material variants of the *Evolute* attribute of both positional Conditionality variants. These are Particular and Indeterminate. There are two distinct material variants of the *Adject* attribute of both Conditionality variants. These are Individual and Relational. There are two alternative extensional forms of the *Adject* attribute. These are Univalent and Multivalent. Facts of the Univalent forms of any variant can be in one of two distinct configurations—Nomologous or Autarkic. Facts of the Multivalent forms are necessarily Nomologous. There are two intensional variants of the Nomologous configurations of Subvenient and Supervenient Conditionality. These are Acception and Exception.

Any minimally correspondent and capable product representation system must implement explicit representations of these Conditionality variants, forms, and configurations as subtypes of the QUALIFICATION relation type and provide facilities for differentiating, instantiating, modifying, and querying them. Figure 31 in Attachment 2 graphically depicts these structures in some detail.

1. ANTECEDENCE

Antecedence relations are structures of the form:

$$\langle \text{Antecedence}, \langle \text{Superject}, \text{Inceptor}, [\text{Antecedent} := \text{Antecedent} \vee \langle \text{Antecedent}_1, \dots, \text{Antecedent}_n \rangle \vee \circ] \rangle \rangle$$

where *Antecedence* designates the Subvenient variant of the Conditionality taxon; where *Superject* designates contingently operant entities; where *Inceptor* designates an Inception phase TRANSFORMATION instance in the ontogeny of the superject entity; and where *Antecedent_i* designates a causal prerequisite for operance of the superject entity.

ANTECEDENCE instances with COMPOSITION, CONSTITUTION, or TRANSFORMATION instances in their *Superject* positions are formalizations of component, raw material, or version “effectivities” in BOMS. Specifically, ANTECEDENCE³⁸ instances (“Q”) with THETIC (“?”) rank TAXON COMPOSITION instances in their *Superject* positions and at least one relation instance an *Antecedent_i* position are formalizations of a diverse class of phenomena commonly called “structure effectivity,” a specific and very common form of intrinsic variation—namely, *compositional contingency*. Such modalized COMPOSITION instances represent *conditional* Containment relationships; that is, relationships between complexes and their components that are only *operant* (“effective”) under certain specific conditions. For example, the COMPOSITION instance in the structure

$$\begin{aligned} & \langle [], \langle \text{?}, \langle \text{F-18}, \text{ElectronicCounterMeasuresPod} \rangle \rangle \rangle \\ & \quad \vdots \\ & \quad \langle \text{Q}, \langle \text{?SOA}, \text{?}, \text{RadarSupressionFunction} \rangle \rangle \\ & \quad \vdots \end{aligned}$$

³⁸ This relation type is defined in paragraph 3.2.2.4 below.

reopresents the fact that members of the F-18 taxon stand in COMPOSITION relations to members of the ElectronicCounterMeasuresPod taxon.³⁹ However, the presence of the modalizing ANTECEDENCE instance marks this relationship as *contingent*. That is, this Containment relationship is *not* canonically operant or invariant among all members of the F-18 and ECMPod taxa, and, accordingly, not all F-18s contain ECM pods as components. A particular F-18 will contain an ECM pod if and only if the RadarSuppressionFunction element of the F-18 functional architecture has been invoked— it is to perform an EW role in the context of a mission.

2. CONSEQUENCE

Consequence relations are *converses* of ANTECEDENCE relations, and are structures of the form:

$$\langle \text{Consequence}, \langle \text{Subject}, \text{Continuance}, [\text{Consequent} := \text{Consequent} \vee \langle \text{Consequent}_1, \dots, \text{Consequent}_n \rangle \vee \circ] \rangle \rangle$$

where *Consequence* designates the Supervenient variant of the Conditionality taxon; where *Subject* designates determinants of contingently operant entities; where *Continuance* designates an Continuation phase TRANSFORMATION instance in the ontogeny of the subject entity; and where *Consequent_i* designates a causal successor of the operance of the subject entity.

3.2.2.5 QUANTIFICATION

Quantification relations are structures of the form:

$$\langle \text{Quantification}, \langle \text{Datum}, \text{Magnitude}, \text{System}, \text{Unit}, \text{Quantity} \rangle \rangle$$

where *Quantification* designates the Multeity taxon; where *Datum* designates entities that fall under a quantificational scope; where *Magnitude* designates specific variants of the Multeity taxon; where *System* designates a mensuration scheme delimiting Multeity taxon variants; *Unit* designates a discrete divisionalization scheme for one or more magnitudes; and where *Quantity* designates a particular number of units. Examples of quantification relations are:

$$\langle \#, \langle \rightarrow, \langle \lambda, \langle \text{C-130}, \text{Engine} \rangle \rangle, 2 \rangle \rangle, \text{Containment}, \text{Unitary}, 4 \rangle \rangle$$

$$\langle \#, \langle \rightarrow, \langle \perp, \langle \text{Water}, \text{Hydrogen} \rangle \rangle, 2 \rangle \rangle, \text{Substance}, \text{SI}, \text{Mole}, 2 \rangle \rangle$$

$$\langle \#, \langle \rightarrow, \langle \odot, \langle \text{PartT}_x, \text{Hole}_y, \odot \rangle \rangle, 2 \rangle \rangle, \text{Cylindricity}, \text{ANSI Y14.5}, \text{Inches}, .005 \rangle \rangle$$

$$\langle \#, \langle \lambda, \langle \text{C-130}, \text{AftPluf} \rangle \rangle, \text{Location}, \text{C-130Grid}, \text{StationLine}, 158 \rangle \rangle$$

where the symbols “#”, “→”, “λ”, “⊥”, and “⊙” respectively denote QUANTIFICATION, INDEXICALIZATION, COMPOSITION, CONSTITUTION, and INHERENCE as noted on page 33.

QUANTIFICATION⁴⁰ instances (“#”) with *Datum* positions containing indexicals (“→”) to *Element* positions in rank TAXON COMPOSITION instances are formalizations of “where used quantities,” a pervasive and essential characteristic of Containment represented by parts lists on engineering drawings and by BOMS. Thus the soas in the structure

$$\begin{array}{l} \langle [], \langle \lambda, \langle \text{C-130}, \text{AllisonTurbopropType}_x \rangle \rangle \rangle \\ \vdots \\ \langle \#, \langle \rightarrow, \langle \lambda \text{SOA}, 2 \rangle \rangle, \text{Multiplicity}, \text{MFS}, \text{Individual}, 4 \rangle \rangle \\ \vdots \end{array}$$

represent the facts that members of the C-130 taxon stand in COMPOSITION relations to members of the AllisonTurbopropType_x taxon, and that, via the QUANTIFICATION instance modalizing that COMPOSITION instance, every member of the former contains exactly 4 members of the latter—in other words, that each c-130 contains four Allison turboprop engines as component elements.⁴¹ Any minimally correspondent and capable product representation system must provide facilities for instantiating,

³⁹ The F-18 airframe contains a centerline pylon that is used to attach ordnance and certain mission-specific hardware to the airplane. The ECM pod, which is basically an elongated bullet-shaped tank stuffed full of radar jamming and other EW equipment, is a case in point.

⁴⁰ This relation type is defined in paragraph 3.2.2.5 below.

modifying, and querying quantifications of COMPOSITION instances, regardless of metasystematic rank.⁴²

QUANTIFICATION instances with Datum positions containing indexicals to Substrate positions in rank TAXON CONSTITUTION instances are formalizations of “make from quantities,” a pervasive and essential characteristic of Materiality represented by materials lists on engineering drawings and by BOMS. Thus the soas in the structure

```

⟨[ ], ⟨1,⟨MountingBracketx,MIL-S-6758/4130⟩⟩⟩
...
⟨#,<⟨→,⟨1SOA,2⟩⟩,Mass,SI,Gram,50⟩⟩
...

```

represent the facts that members of the MountingBracket_x taxon stand in CONSTITUTION relations to members of the 4130 variant of the MIL-S-6758 taxon (i.e., chrome steel), and that, via the QUANTIFICATION instance modalizing that CONSTITUTION instance, every member of the former is made out of 50 grams of the latter—in other words, that each individual mounting bracket of that type contains four Allison turboprop engines as elements.⁴³ Any minimally correspondent and capable product representation system must provide facilities for instantiating, modifying, and querying quantifications of rank TAXON COMPOSITION instances.

3.2.3 Intrinsic Adjunct Metastructures

	TYPE	RELATION OF	RELATION BETWEEN↔	EXAMPLES
ADJUNCT INTRINSIC	6 EQUIVALENCE	<i>Substitutionality</i>	Relata↔Substituends	MIL-P-18177 TYPE G10↔LP-509 CLASS II GRADE A
	7 ALTERNATION	<i>Optionality</i>	Relata↔Alternatives	F-16 AVIONICS↔COUNTRY-SPECIFIC PACKAGES
	8 VARIATION	<i>Diversity</i>	Baselines↔Variants	JSF↔⟨AF VARIANT,STOL VARIANT⟩
	9 ORDER	<i>Positionality</i>	Entities↔Positions	ASSEMBLY ORDER; SERVICE PRECEDENCE
	10 TRANSFORMATION	<i>Development</i>	Entities↔States	C-130H↔C-130J; LINUX KERNEL V2.0.3↔V2.0.4

Table 3. Intrinsic Relations

All extrinsic and processual variabilities within the structures of artifacts minimally consist in five fundamental types of relations.

1. EQUIVALENCE —the relation of *substitutionality*;
2. ALTERNATION —the relation of *optionality*;
3. VARIATION —the relation of *diversity*;
4. ORDER —the relation of *positionality*; and,
5. EVOLUTION —the relation of *change*.

Any minimally correspondent product representation system must implement representations of these five relation types and provide facilities for instantiating, modifying, and querying them. Examples of entities standing in these relations are depicted in the table at the top of Figure 33 in Attachment 2.

⁴¹ The actual matters of fact are of course much more complicated than those represented by this simplistic example. For instance, c-130's in the process of being built don't contain even *one* engine (let alone 4) until they reach a certain stage of assembly, so the QUANTIFICATION instance itself should be qualified by an ANTECEDENCE instance representing this effectivity.

⁴² Quantifications of rank INDIVIDUAL COMPOSITION instances are frequently employed to represent particular but individually variable numbers of components in serialized “AS-BUILT” and “AS-REPAIRED” BOMS.

⁴³ The actual matters of fact are of course much more complicated than those represented by this simple-minded example. For instance, c-130's in the process of being built don't contain even *one* engine (let alone 4) until they reach a certain stage of assembly, so the QUALIFICATION instance should itself be qualified by an ANTECEDENCE instance representing its effectivity.

3.2.3.1 EQUIVALENCE

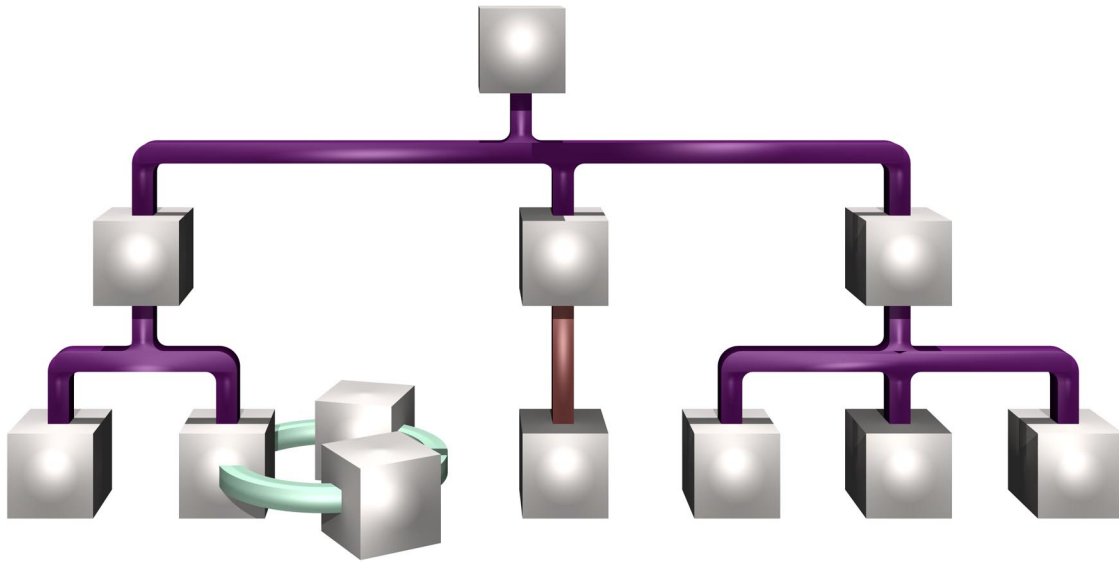


Figure 5. Equivalence Relations

Equivalence relations, depicted by the circular ring of green lines in Figure 5 below, are structures of the form:

$\langle \text{Equivalence}, \langle \text{Substituend}_1, \text{Substituend}_2, \dots, \text{Substituend}_n \rangle \rangle$

where *Equivalence* designates the Substitutability taxon; and where *Substituend_i* designates an entity that is completely *interchangeable* with at least one other entity.

3.2.3.2 ALTERNATION

Alternation relations, depicted by the blue lines bottom left in the Figure 6 below, are structures of the form:

$\langle \text{Alternation}, \langle \text{Context}, \langle \text{Alternant}_1, \text{Alternant}_2, \dots, \text{Alternant}_n \rangle \rangle \rangle$

where *Alternation* designates the Optionality taxon; where *Context* designates entities situating the alternation; and where *Alternant_i* designates an entity that is form, fit, function *congruent* with at least one other entity.

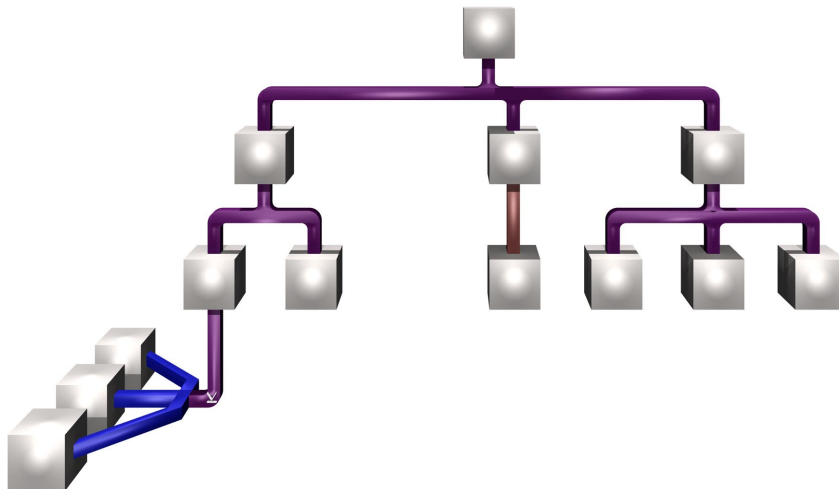


Figure 6. Alternation Relations

3.2.3.3 VARIATION

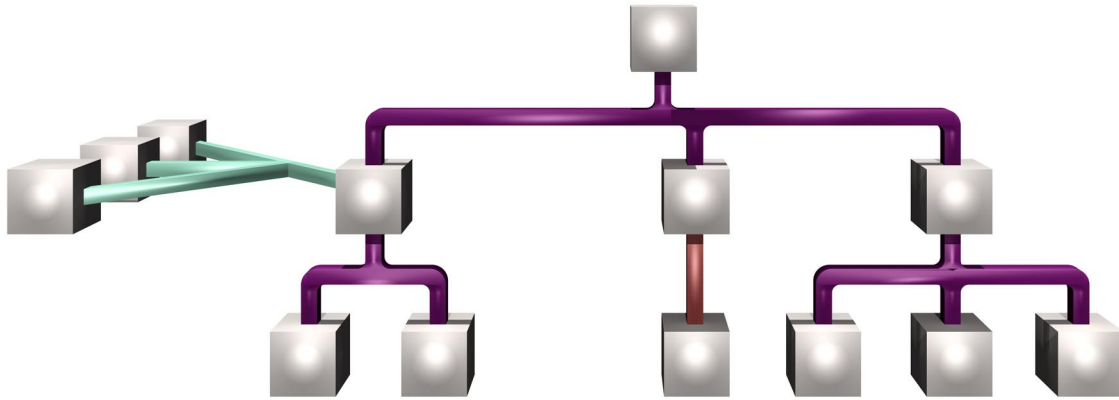


Figure 7. Variation Relations

Variation relations, depicted by the green lines upper left in the above figure, are structures of the form:

$$\langle \text{Variation}, \langle \text{Baseline}, \langle \text{Divergence}_1, \dots, \text{Divergence}_n \rangle, \text{Variant} \rangle \rangle$$

where *Variation* designates the Intraspecific Variety taxon; where *Baseline* designates entities that are progenitor or canonical taxonomic configurations; where *Variant* designates entities that are derivative configurations of progenitor or canonical configurations; and where *Divergence_i* designates an additive or subtractive *delta* between variants and their respective baseline configurations.

3.2.3.4 ORDER

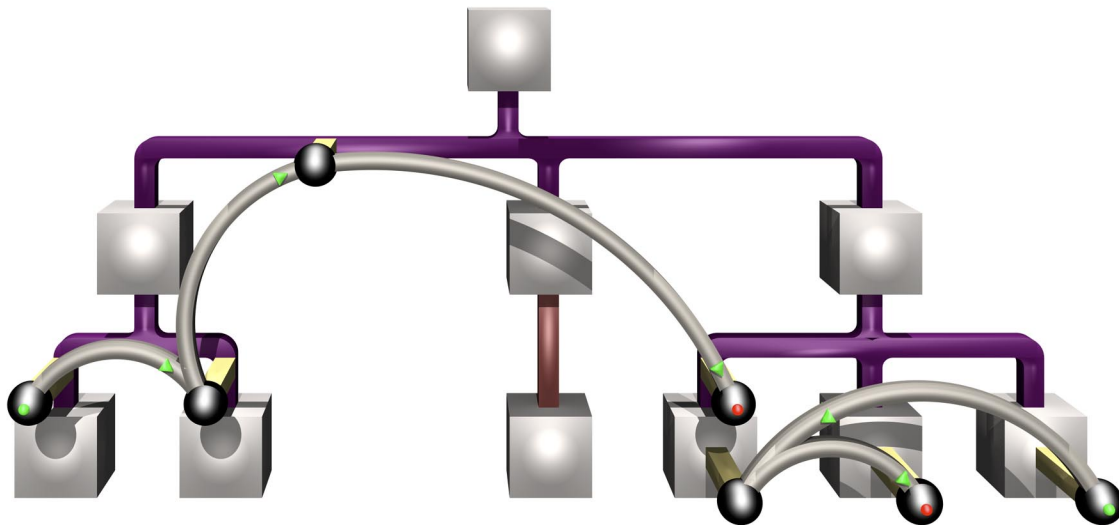


Figure 8. Order Relations

Order relations, depicted by the curved grey lines in the above figure, are structures of the form:

$$\langle \text{Order}, \langle \text{Ordinate}, [\text{Ordinal} := \text{Ordinal} \vee \langle \text{Ordinal}_1, \dots, \text{Ordinal}_n \rangle \vee \circ] \rangle \rangle$$

where *Order* designates the Positionality taxon; *Ordinate* designates entities standing in some relative position with respect to others; where *Ordinal_i* designates an instance of the ORDER relation type; and where \circ designates an instance of the NULL PARTICULAR variant of IDENTITY.

I ORDER Variants

There are two major extensional variants of Positionality—Contiguous and Furcate. There are two principal alternative forms of Contiguous Positionality. These are Consecutive and Concurrent. There are two principal alternative forms of Furcate Positionality. These are Excessional and Ingressive. Facts of the Consecutive form of the Contiguous variant and the Ingressive form of the Furcate variant can be in one of two configurations—Catenary and Terminal. Any minimally correspondent and capable product representation system must accordingly implement explicit representations of these four variants and two configurations as subtypes of the ORDER relation type and provide facilities for instantiating, modifying, and querying them.

1. SEQUENCE

Sequence relations are structures of the form:

$$\langle \text{Sequence}, \langle \text{Ordinate}, [\text{Ordinal} \equiv \text{Successor} \vee \circ] \rangle \rangle$$

where **Sequence** designates the Consecutive form of the Contiguous variant of the Positionality taxon; where **Ordinate** designates entities standing in a position within a contiguous series; and where **Successor** designates an instance of the ORDER relation type. **SEQUENCE** instances containing ORDER relation instances in the **Ordinal** attribute position represent facts related to proximate facts in an ordering. Those with **NULL** particulars in that position represent facts terminating an ordering. Examples of **SEQUENCE** relations are:

$$\begin{array}{ll} \text{Seq}_i: \langle \text{Sequence}, \langle \langle \text{Composition}, \langle X, Y \rangle \rangle, \text{Seq}_i \rangle \rangle & \\ \text{Seq}_j: \langle \text{Sequence}, \langle \langle \text{Composition}, \langle X, Z \rangle \rangle, \circ \rangle \rangle & \text{(e.g., assembly ordering for X);} \\ \hline \text{Seq}_i: \langle \text{Sequence}, \langle \text{Process}_q, \text{Seq}_i \rangle \rangle & \\ \text{Seq}_j: \langle \text{Sequence}, \langle \text{Process}_w, \text{Seq}_k \rangle \rangle & \text{(e.g., process execution ordering);} \\ \hline \text{Seq}_i: \langle \text{Sequence}, \langle \langle \text{Composition}, \langle X, Y \rangle \rangle, \text{Seq}_i \rangle \rangle & \\ \text{Seq}_j: \langle \text{Sequence}, \langle \langle \text{Composition}, \langle X, Z \rangle \rangle, \circ \rangle \rangle & \text{(e.g., assembly ordering for X);} \end{array}$$

2. COINCIDENCE

Coincidence relations are structures of the form:

$$\langle \text{Coincidence}, \langle \text{Ordinate}, \langle \text{Coincident}_1, \text{Coincident}_2, \dots, \text{Coincident}_n \rangle \rangle \rangle$$

where **Coincidence** designates the Concurrent form of the Contiguous variant of the Positionality taxon; **Ordinate** designates entities that are in a parallel position with others in a contiguous series; and where **Coincident_i** designates an instance of the ORDER relation type.

3. RAMIFICATION

Ramification relations are structures of the form:

$$\langle \text{Ramification}, \langle \text{Ordinate}, \langle \text{Disjunct}_1, \text{Disjunct}_2, \dots, \text{Disjunct}_n \rangle \rangle \rangle$$

where **Ramification** designates the Excessional form of the Furcate variant of the Positionality taxon; **Ordinate** designates entities that are incident to the same position in a contiguous series; and where **Disjunct_i** designates an instance of the ORDER relation type.

4. CONVERGENCE

Convergence relations are structures of the form:

$$\langle \text{Convergence}, \langle \text{Ordinate}, \langle \text{Precessor}_1, \text{Precessor}_2, \dots, \text{Precessor}_n \rangle, [\text{Successor} \vee \circ] \rangle \rangle$$

where **Convergence** designates the Ingressional form of the Furcate variant of the Positionality taxon; **Ordinate** designates entities that are incident to the same position in a contiguous series; and where **Precessor_i** and **Successor** designate instances of the ORDER relation type.

3.2.3.5 TRANSFORMATION

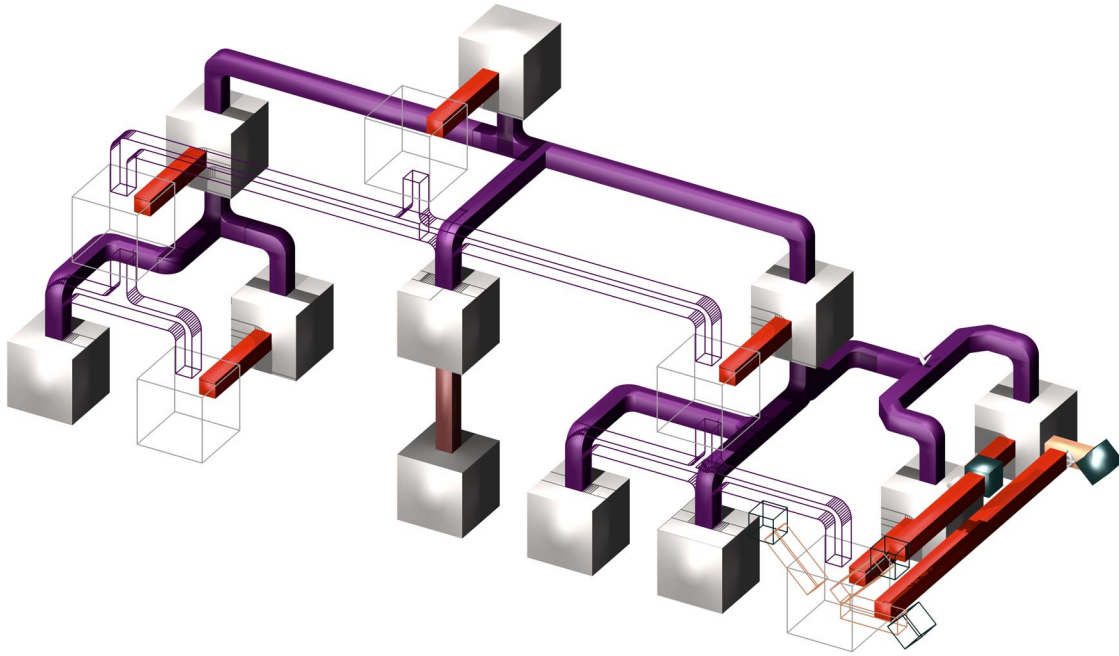


Figure 9. Evolution Relations

Transformation relations, depicted by the red lines in the above figure, are structures of the form:

$$\langle \text{Transformation}, \langle \text{Subject}, [\text{Delta} := \text{Delta} \vee \circ], [\text{Evolute} := \text{Evolute} \vee \langle \text{Evolute}_1, \dots, \text{Evolute}_n \rangle \vee \circ] \rangle \rangle$$

where Transformation designates the Development taxon; where Delta designates relations of additive or subtractive change; where Evolute_i designates an instance of the TRANSFORMATION relation type; and where NULL designates an instance of the NULL PARTICULAR variant of IDENTITY.

Transformation is, in actuality, a variant of the ORDER metatype. That is, transformation is *positionality* of *change*, as opposed to any other ordinate type. It is defined here as a distinct metasystematic peer of ORDER because of its central role in delimiting the structures of ontogeny—i.e., *versioning*.

3.2.4 Extrinsic Adjunct Metastructures

		TYPE	RELATION OF	RELATION BETWEEN↔	EXAMPLES
ADJUNCT EXTRINSIC	11	ARTICULATION	<i>Separation</i>	Entities↔Realizations	NOZZLE ASSEMBLY↔{THROAT,CHAMBER,EXIT CONE}
	12	FACTORIZATION	<i>Abstraction</i>		⟨GEAR ₁ ,GEAR ₁ ,GEAR ₁ ⟩↔GEAR BLANK
	13	CONSOLIDATION	<i>Integration</i>		⟨CHIPS,BOARDS,HOUSING⟩↔AVIONICS LRU

Table 4. Extrinsic Relations

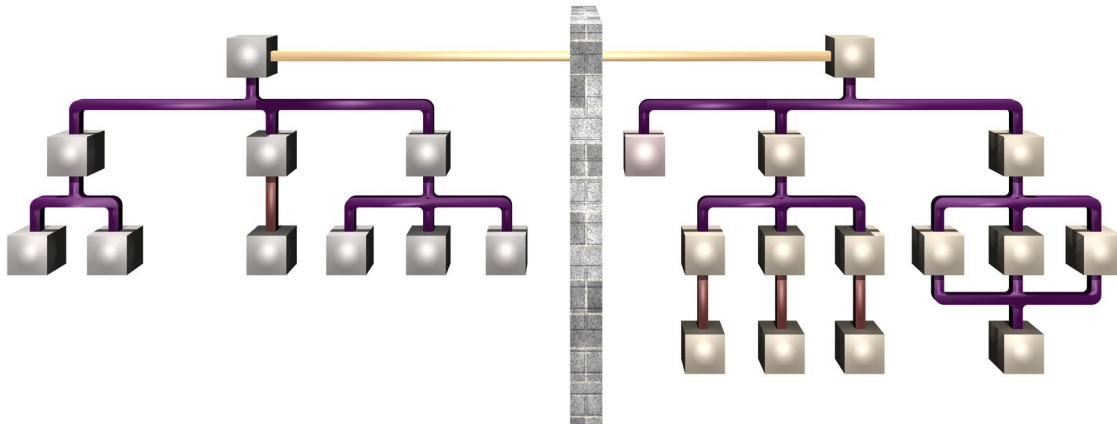


Figure 10. Extrinsic Relations

All divergences between multiple genitive artifact configurations minimally consist in three fundamental types of relations.

1. ARTICULATION —the relation of *separation*;
2. FACTORIZATION —the relation of *abstraction*; and,
3. CONSOLIDATION —the relation of *integration*.

Any minimally correspondent product representation system must accordingly implement explicit representations of these three relation types and provide facilities for instantiating, modifying, and querying them. Examples of entities standing in these relations are depicted in the table at the top of Figure 33 in Attachment 2. A detailed discussion of them is presented in Simons and Dement [21].

3.2.4.1 ARTICULATION

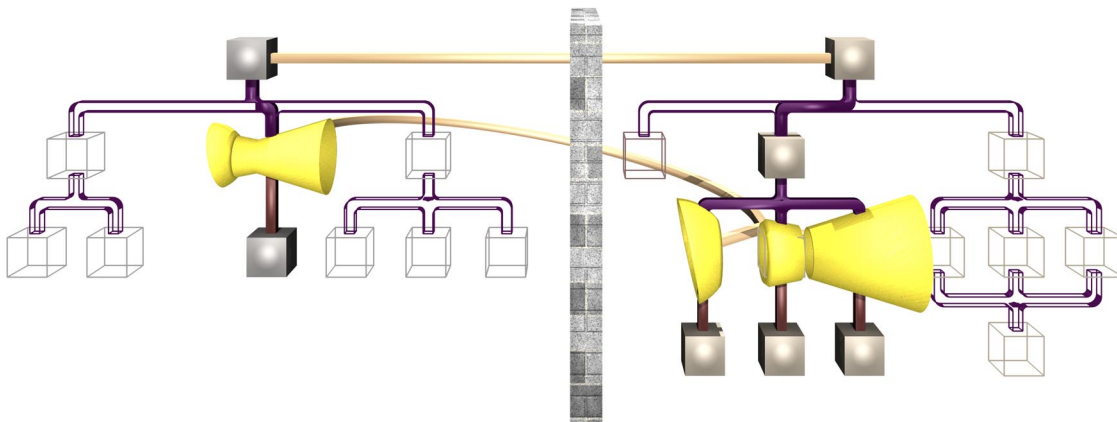


Figure 11. Articulation Relations

Articulation relations, depicted by the orange lines between the rocket nozzle image on the left

and right in the Figure 11, are structures of the form:

$$\langle \text{Articulation}, \langle \text{Choate}, [\text{Intrastice} := \text{Intrastice} \vee \langle \text{Intrastice}_1, \dots, \text{Intrastice}_n \rangle], \langle \text{Segment}_1, \text{Segment}_2, \dots, \text{Segment}_n \rangle \rangle \rangle$$

where Articulation designates the Separation variant of the Transposition taxon; where Choate designates unitary elements of a particular choate entity configuration; where Intrastice_i designates an entity constituting an actual or potential processual interdiction with respect to a second configuration of a given choate entity configuration; and where Segment_i designates an entity constituting a partition of the choate entity in the second configuration.

3.2.4.2 FACTORIZATION

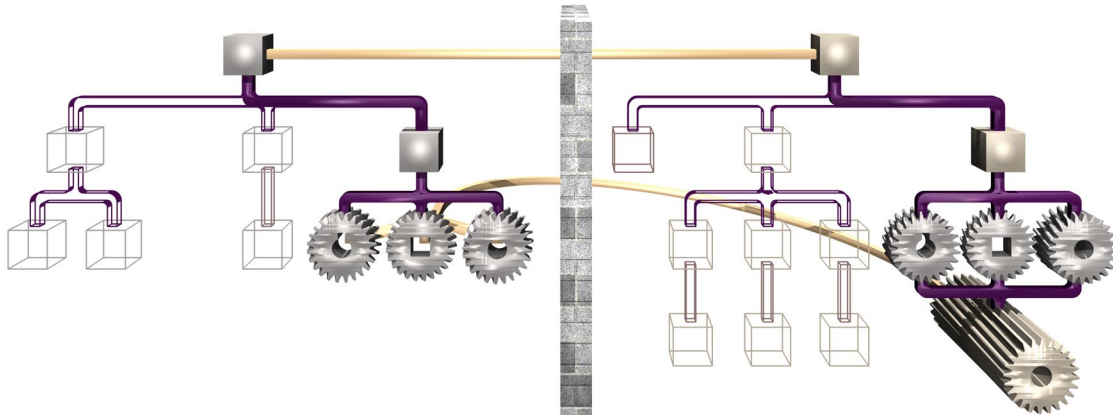


Figure 12. Factorization Relations

Factorization relations, depicted by the orange lines between the gears on the left and the gear blank on the right in the Figure 12, are structures of the form:

$$\langle \text{Factorization}, \langle \langle \text{Cognate}_1, \text{Cognate}_2, \dots, \text{Cognate}_n \rangle, [\text{Cœnomorph} := \text{Cœnomorph} \vee \langle \text{Cœnomorph}_1, \dots, \text{Cœnomorph}_n \rangle], \text{Factor} \rangle \rangle$$

where Factorization designates the Abstraction variant of the Transposition taxon; where Cognate_i designates an entity that is phyletically related to other elements of a particular entity configuration; where Cœnomorph_i designates a shared character among cognate entities; and where Factor designates an entity constituting a coincident bearer of one or more cœnomorphs in a second configuration of a given entity.

3.2.4.3 CONSOLIDATION

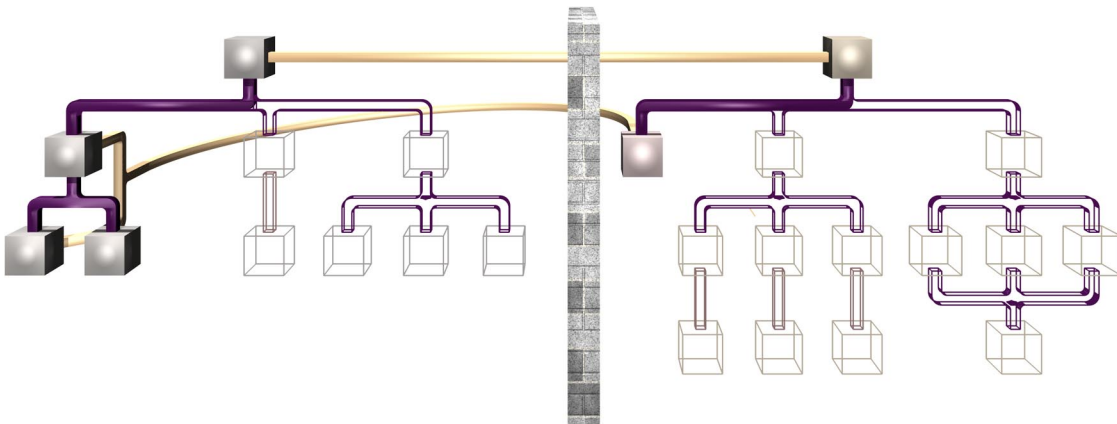


Figure 13. Consolidation Relations

Consolidation relations, depicted by the orange lines between the three elements on the left and the one on the right in Figure 13, are structures of the form:

$\langle \text{Consolidation}, \langle \langle \text{Isolate}_1, \text{Isolate}_2, \dots, \text{Isolate}_n \rangle, [\text{Interstice} := \text{Interstice} \cup \langle \text{Interstice}_1, \dots, \text{Interstice}_n \rangle], \text{Combinant} \rangle \rangle$

where Consolidation designates the Abstraction variant of the Transposition taxon; where Isolate_i designates an entity that is discontinuous with other elements of a particular entity configuration; where Interstice_i designates separation between discontinuous entities; and where Combinant designates an entity constituting an integral entity encompassing discontinuous entities in a second configuration of a given entity.

3.2.5 Intertypic Adjunct Metastructures

		TYPE	RELATION OF	RELATION BETWEEN↔	EXAMPLES
ADJUNCT INTERTYPIC	14	INVOLVEMENT	<i>Participation</i>	Continuants↔Occurs	PRODUCTS↔FUNCTIONS; ATTRIBUTES↔TEST PROCS
	15	CAPACITATION	<i>Provisionment</i>	Involvements↔Tools; Agencies	F-18E/F ASSEMBLY↔{FIXTURES; JIGS; ASSEMBLERS}
	16	REPRESENTATION	<i>Objectification</i>	Entities↔Representations	PRDCTS↔{ENG DWGS, DATA}; PROCS↔{TECH MNLS, CODE}
	17	DESIGNATION	<i>Nominalization</i>	Entities↔Designators	PARTS↔PART NUMBERS; AIRCRAFT↔TAIL NUMBERS

Table 5. Intertypic Relations

All structures of all artifacts minimally consist in five fundamental types of relations.

1. INVOLVEMENT —the relation of *participation*;
2. CAPACITATION —the relation of *provisionment*;
3. REPRESENTATION—the relation of *objectification*; and,
4. DESIGNATION —the relation of *nominalization*.

Any minimally correspondent product representation system must accordingly implement explicit representations of these four relation types and provide facilities for instantiating, modifying, and querying them. Examples of entities standing in these relations are depicted in the table at the top of Figure 33 in Attachment 2.

3.2.5.1 INVOLVEMENT

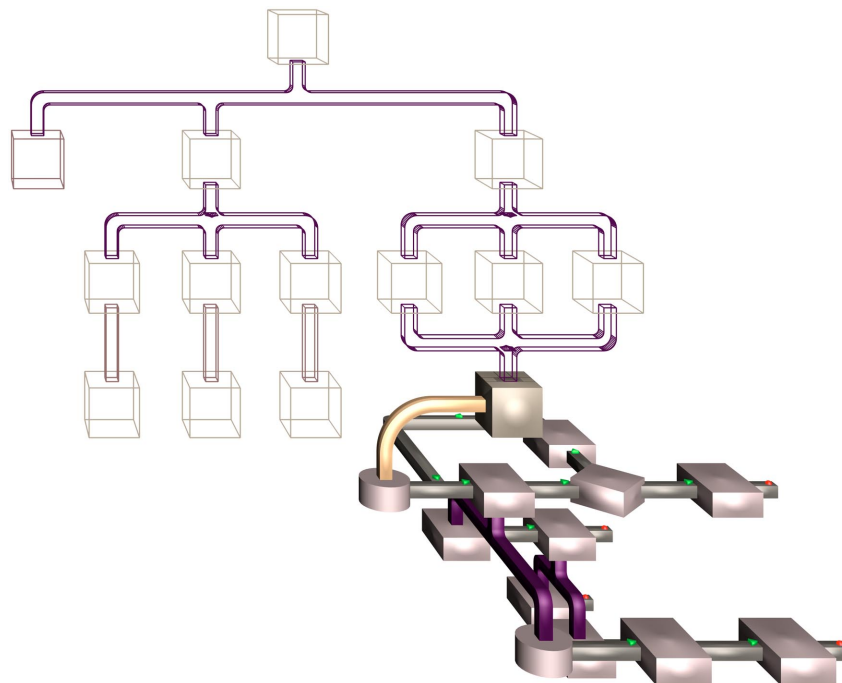


Figure 14. Involvement Relations

Involvement relations, depicted by the orange ‘elbow’ line in Figure 14, are structures of the form:

$\langle \text{Involvement}, \langle \text{Type}, \text{Intension}, \text{Extension} \rangle \rangle$

where *Involvement* designates the Participation taxon; where *Type* designates taxa in a given representation metasystem; where *Intension* designates entities constituting context elements of facts of representation system types; and where *Extension* designates entities constituting elements in the extants of those facts.

3.2.5.2 CAPACITATION

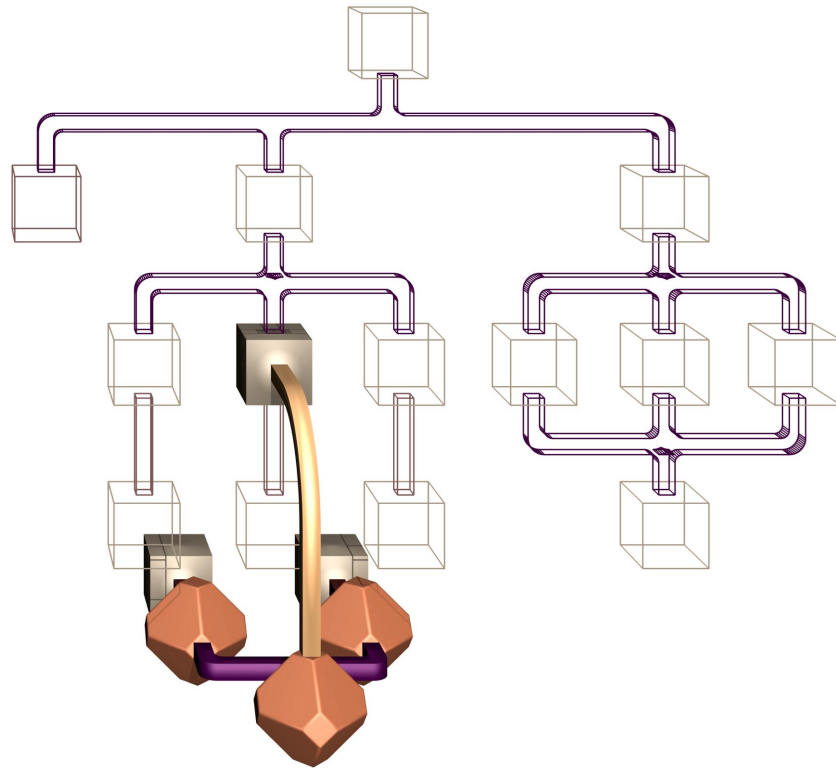


Figure 15. Capacitation Relations

Capacitation relations, depicted by the orange arc line in Figure 15, are structures of the form:

$\langle \text{Capacitation}, \langle \text{Involvement}, \text{Means}, \text{Yield} \rangle \rangle$

where *Capacitation* designates the Provisionment taxon; *Involvement* designates instances of the INVOLVEMENT relation type as defined in paragraph 3.2.5.1 above; *Means* designates entities comprising implemental mechanisms or agencies; and where *Yield* designates entities constituting actualizations of *antecedents* (conditions or inputs) of INVOLVEMENT relation instances.

3.2.5.3 REPRESENTATION

Representation relations, depicted by the orange lines in the Figure 16, are structures of the form:

$\langle \text{Representation}, \langle \text{Concept}, \text{Content}, \text{Object} \rangle \rangle$

where *Representation* designates the Objectification taxon; where *Concept* designates taxa delimiting facts of intentional content; where *Content* designates objectified facts—entities that are representations of entities; and where *Object* designates represented entities. A detailed discussion of REPRESENTATION is presented in Smith [22].

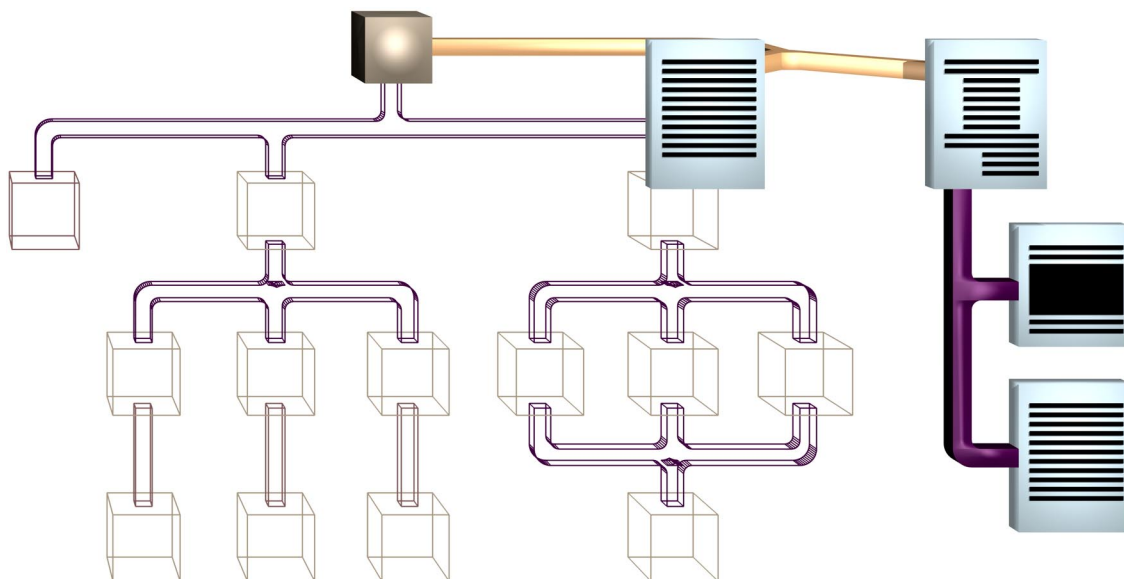


Figure 16. Representation Relations

3.2.5.4 DESIGNATION

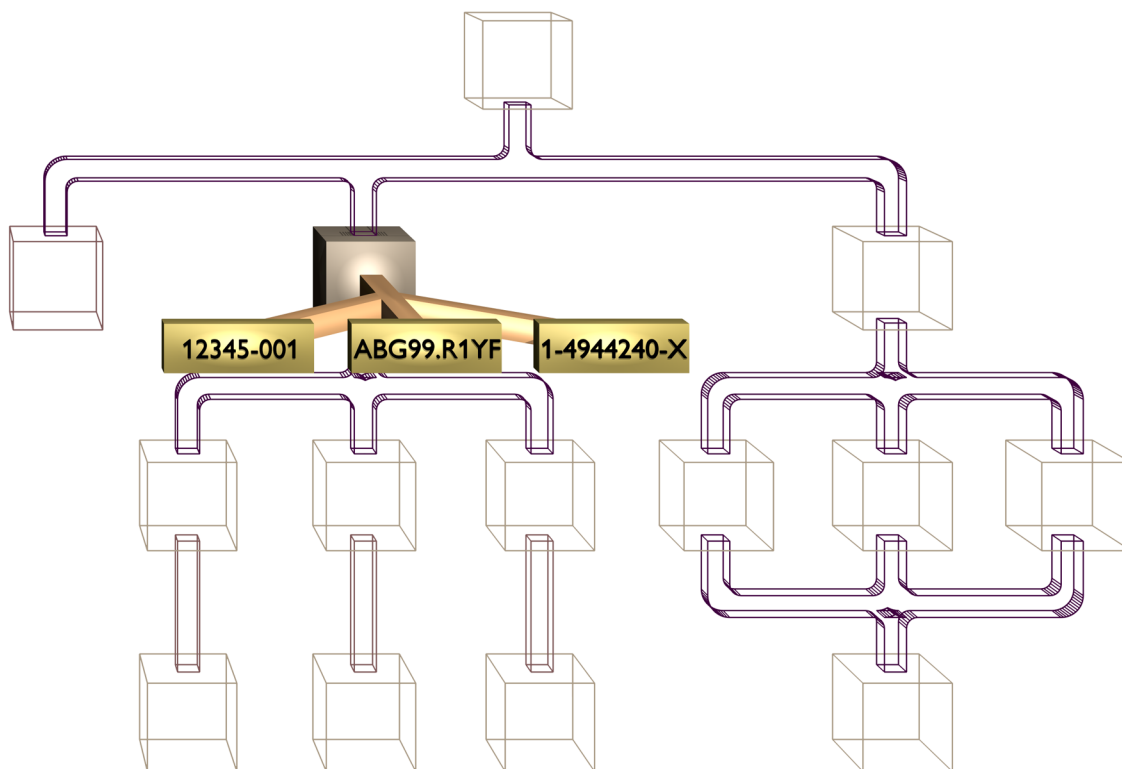


Figure 17. Designation Relations

Designation relations, depicted by the orange lines in the above figure, are structures of the form:

$$\langle \text{Designation}, \langle \text{Context}, \text{Designator}, \text{Referent} \rangle \rangle$$

where **Designation** designates the Nominalization variant of the Objectification taxon; **Context** design-

nates taxa delimiting nominal facts; where Designator designates entities that are denominations of entities; and where Referent designates denominated entities.

3.2.6 Table of Metastructure Schemata

Table 6. Metastructure Schemata

⟨Composition,⟨Complex,[Element := Element \vee \circ] ⟩⟩
⟨Constitution,⟨Superstrate,[Substrate := Substrate \vee \circ] ⟩⟩
⟨Inherence,⟨Bearer,Character,[Basis := Basis \vee \circ] ⟩⟩
⟨Qualification,⟨Correlate,[Evolute := Evolute \vee \circ],[Adject := Adject \vee ⟨Adject ₁ ,...,Adject _n ⟩ \vee \circ] ⟩⟩
⟨Antecedence,⟨Superject,Inceptor,[Antecedent := Antecedent \vee ⟨Antecedent ₁ ,...,Antecedent _n ⟩ \vee \circ] ⟩⟩
⟨Consequence,⟨Subject,Continuance,[Consequent := Consequent \vee ⟨Consequent ₁ ,...,Consequent _n ⟩ \vee \circ] ⟩⟩
⟨Quantification,⟨Datum,Magnitude,System,Unit, [Value:= Value \vee Range \vee SoA] ⟩⟩
⟨Equivalence,⟨Substituend ₁ ,Substituend ₂ ,... _n ⟩⟩
⟨Alternation,⟨Context,⟨Alternant ₁ ,Alternant ₂ ,... _n ⟩⟩⟩
⟨Variation,⟨Baseline,⟨Divergence ₁ ,...,Divergence _n ⟩,Variant⟩⟩
⟨Order,⟨Ordinate,[Ordinal := Ordinal \vee ⟨Ordinal ₁ ,...,Ordinal _n ⟩ \vee \circ] ⟩⟩
⟨Sequence,⟨Ordinate,[Ordinal := Successor \vee \circ] ⟩⟩
⟨Coincidence,⟨Ordinate,⟨Coincident ₁ ,Coincident ₂ ,... _n ⟩⟩⟩
⟨Ramification,⟨Ordinate,⟨Disjunct ₁ ,Disjunct ₂ ,...,Disjunct _n ⟩⟩⟩
⟨Convergence,⟨Ordinate,⟨Precessor ₁ ,Precessor ₂ ,... _n ⟩,[Successor \vee \circ] ⟩⟩
⟨Transformation,⟨Subject,[Delta := Delta \vee \circ],[Evolute := Evolute \vee ⟨Evolute ₁ ,...,Evolute _n ⟩ \vee \circ] ⟩⟩
⟨Articulation,⟨Individual,[Intrastice := Intrastice \vee ⟨Intrastice ₁ ,...,Intrastice _n],⟨Moiety ₁ ,Moiety ₂ ,... _n ⟩⟩⟩
⟨Factorization,⟨⟨Cognate ₁ ,Cognate ₂ ,... _n ⟩,[Cœnomorph := Cœnomorph \vee ⟨Cœnomorph ₁ ,...,Cœnomorph _n ⟩],Factor⟩⟩
⟨Consolidation,⟨⟨Isolate ₁ ,Isolate ₂ ,... _n ⟩,[Interstice := Interstice \vee ⟨Interstice ₁ ,...,Interstice _n ⟩],Combinant ⟩⟩
⟨Involvement,⟨Type,Intension,Extension⟩⟩
⟨Capacitation,⟨Involvement,Means,Yield⟩⟩
⟨Application,⟨Involvement,Device,Effect⟩⟩
⟨Instrumentalization,⟨Involvement,Agency,Solution⟩⟩
⟨Representation,⟨Concept,Content,Object⟩⟩
⟨Designation,⟨Context,Designator,Referent⟩⟩

“We shape our buildings: thereafter they shape us.”

Winston Churchill, 1962

4

RESULTS PART II: *Enterprise Operating System Definition*

After nearly a year of planning and preparation, a program to ‘re-invent’ LMAS,⁴⁴ called the LMAS Enterprise Architecture Program (LEAP), was launched in May of 1999. The aim of this program was to enhance LMAS competitiveness by deploying a new enterprise operating system for the company. The specific objectives were to establish a formal enterprise Strategy process; to re-design the ‘core value stream’ processes (i.e., Product Realization and its major subprocesses); to integrate these two; and to streamline their respective organizations and infrastructure.

Our prior involvement in planning for this program, together with our development of a solution to the multiple BOM reconciliation and other related product representation problems, had equipped us to play a unique formalist role in this activity. Specifically, one major element of the LEAP approach involved pioneering the use of the new formal methods and metascystematics we had developed under MEREOS phases I and II to design new enterprise processes for LMAS. Hence in accordance with an agreement among all principal MEREOS program stakeholders, we were brought into the program as architects, and the principal emphasis of our statement of work in MEREOS Phase III was re-focused to accommodate that role.⁴⁵ Our specific mission was to develop *formal* structures and *schematic* content to frame, inform, and ‘vector’ the operating system design and deployment activities. That is, we were primarily focused on formal operating system requirement definitions, and on insuring that these were rigorous and systematic, complete, and traceable to stated leadership intents—the attributes necessary to render formal definitions of use in vectoring and assessing specific designs. In other words, most of the functions we were tasked to perform were analogous to those performed by a systems engineering group in a product development program. Given that the ‘product’ under ‘development’ was an enterprise operating system, one might suggestively label these as *enterprise engineering*.

⁴⁴ Lockheed Martin Aeronautical Systems, Marietta, GA.

⁴⁵ The technical factors and strategic circumstances that brought about our involvement in LEAP were presented in paragraph 2.6 of the Introduction.

By late 1999, a corporate decision had been taken to consolidate the four LM aeronautical sector companies into a single company headquartered at Ft. Worth—LM Aero. This decision rendered LEAP unnecessary, and the program was accordingly terminated. However, the aim and specific objectives of this consolidation were, for all practical purposes, the same as those of LEAP, albeit in a sector-wide rather than a single company context. And although LEAP was incomplete when it was overtaken by events, a great deal of work had been accomplished, and a great many lessons had been learned. Because of these two factors we were asked to continue in essentially the same formalist capacity as before to support the ‘enterprise engineering’ activity of consolidation program.

The utility of formalist activities, for example those such as ours in LEAP and later in the LM Aero consolidation, and the value of their intrinsically abstract products are notoriously difficult to clearly articulate, especially to those responsible for delivering concrete results. Nevertheless, an understanding the formalist role in concrete endeavors is important for understanding the results we present here.

It is a practical matter of fact that organizations never have the resources or time required to devise, let alone deploy, perfect responses to the challenges and opportunities that confront them. Cogent actions and effective products in the real world always embody constraint-mediated balances between what is possible in principle and what is adequate in practice to meet the needs at hand. The essential role of practitioners is to use their *intuition*, sustained and governed by their experience, to strike such balances. That is, to determine the most realistically attainable proportions of possibility and sufficiency that is feasible given the operant constraints—and, thereby, to choose which actions will be taken among the available alternatives, and to consequently shape the solutions that will be created.

But novel, complex, and risk-intensive endeavors can overtax the experience and intuitive skills of even the most gifted of practitioners. And a program to design and deploy a new operating system for a major aerospace and defense contractor is most definitely a distinctive, very complicated, and risk-laden endeavor. The essential role of formalists in such activities is to *provision* practitioners with tools to *augment* their intuitive experienced-based skills and to *inform* the decisions they must make in these kinds of circumstances. These ‘formal provisions’ invariably consist in at least the following four products. The first are precise and complete *characterizations* of the challenge or opportunity at issue. The second are delimitations ascertaining the *boundaries* of the possible solutions. The third are derivations discerning the *necessary elements* of sufficient solutions, and distinguishing these from elements which are not. The fourth are identifications of appropriate *effectiveness criteria*, both for subsequent actions and for the decisions themselves. Taken together, these schematic tools augment the clarity, precision, and depth of practitioner understanding; they frame the threshold conditions of possible/sufficient balances; and, they pinpoint suitable measures for validating decisions and governing subsequent actions. These tools do not replace experienced judgement or concrete action. But they can when properly exploited significantly enhance a practitioner’s abilities to render effective decisions.

The value we have consistently sought to deliver via our formal enterprise engineering activities should be clearer in light of the preceding remarks. Our mission was to provision cognizant LMAS and LM Aero practitioners with formal tools to facilitate their endeavors to re-design their mechanisms and infrastructure of enterprise action. The approach we adopted to accomplish this was to design an idealized operating system what we call the New Aerospace Enterprise (NAE) operating system, and to iteratively redact and document crucial elements of that design, thereby providing LM Aero practitioners with useful definitions for their concrete purposes that were, nevertheless, explicitly derived from and therefore traceable to what is formally possible.

The activities we performed roughly consisted in needs identification and requirements definitions, the latter encompassing methodology development, formal systematics, mission definition, and schematic designs of enterprise operating system processes. We have organized our presentation of the results of those efforts along these same lines and in this order. Note that while these results complete the MEREOS Phase III sow, they are *interim* products of work in progress in the larger NAE operating system definition context. We are still actively engaged in that activity, and the results described here do not represent the final products of this effort.

4.1 Enterprise Operating System Needs

Enterprises are intentional systems constituting agencies for *rendering solutions* to requirements that ultimately originate from stakeholder purposes. Accordingly, enterprises are *intermediate* instrumentalities for accomplishing those aims—the solutions they produce *provision* stakeholder operations for achieving them.

The primary challenge faced by every enterprise is to attain and sustain success. Success obtains when results of enterprise actions *are* instrumental solutions for operations that accomplish stakeholder purposes. That is, success obtains when there is *effective* instrumentalization *and* application.⁴⁶ Success endures when effective instrumentalization and application are consistently repeatable—i.e., *reliably* effective. Thus, at first glance, success hinges on creating and maintaining alignments between stakeholder purposes and enterprise capabilities to realize solutions to the intents, needs, and requirements deriving from those purposes.

Enterprise capabilities subsist in the capabilities of *enterprise operating systems*—instrumentalities for purposeful collective action. These are integrated 'socio-technical' systems comprising the processes, organizations, policies, and infrastructure (facilities, methods and techniques, information systems, etc.) that provision enterprises to act—to realize solutions that in turn provision accomplishment of stakeholder purposes. The inputs to an enterprise operating system are leadership *intents* and objectives that define and prioritize stakeholder purposes. Its outputs are *solutions* to the needs and requirements pertaining to those purposes. Its means of producing those solutions are enterprise *processes*. Thus success ultimately hinges on creating and maintaining *parity* between action program requirements deriving from enterprise objectives on the one hand, and enterprise operating system capabilities to execute those programs and thereby satisfy those requirements on the other.

Achieving and sustaining this balance is exacting in stable environments and extremely difficult in volatile ones, and business environments are more fluid today than ever. Moreover, increasingly intense competitive pressures driven by factors such as globalization have compressed the time enterprises have to respond to change. Enterprises in every business domain need operating systems that both assure consistent performance *and* rapidly adjust to change in order to succeed and remain competitive.

The end of the Cold War initiated sweeping changes in the defense industry. Two of note were dramatic reductions in the number and scale of weapon system acquisition programs and its consequences: drastic diminution of the business base, collateral downsizing and subsequent consolidation of the defense industrial base, and intensification of competition for what business opportunities remain. Two superficially opposed trends sparked by these changes and by globalization have also emerged. The first is the increasing tendency of governments to vigorously protect their defense industrial infrastructures from extra-national competition and to sustain their capabilities in the face of decreasingly frequent opportunities to exercise them. The second is a marked increase in collaboration among defense contractors, both within and across national boundaries, to pool scarce resources and to exploit complementary capabilities and congruent interests. In the case of the Joint Strike Fighter (JSF) program, this cooperation has extended beyond collaboration among contractors of different nations to include the governments of the nations themselves. Thus sustaining competitiveness in this new aerospace and defense environment entails *new* enterprise operating system capabilities to meet *new* needs, over and above basic effectiveness, reliability, and agility. A sketch of the four most significant of these from an operating system design perspective follows.

4.1.1 Greater Product Realization Amplitude

One major consequence of globalization is that stakeholder populations are both much larger and more diverse in their aims than ever before. The elements constituting defense program stakeholder value are correspondingly more complex and multifaceted. Technical superiority is not the dominating determinant of value it once was. Complex and abstract strategic factors, such as infrastruc-

⁴⁶ An overly technical way of saying that enterprises succeed when their stakeholders do.

ture capability retention and economic asset multiplication, increasingly represent major constituents of value, and the processes of determining and delivering these involves correspondingly more comprehensive enterprise operating system processes.

This in turn entails a major *architectural* change in the relationship of stakeholder value to the Product Realization process. Specifically, the long-standing conception of Product Realization as the principal and dominant enterprise mechanism for delivering value is simply too narrow to encompass the realities of the new defense environment. Consider: the principal components of solutions rendered by Cold War programs were technical systems. That is, if one envisions the *total* solution as a cube, and that element of the solution produced by these programs as another cube, nested inside the first one, the two were typically almost the same size. Hence the total solution and the technical solution were, for all practical purposes, co-extensive, and technical factors were the primary determinants of value. However, this is simply no longer true; the technical JSF solution for example constitutes a much smaller portion of the total JSF solution than did those of any prior program. Success accordingly necessitates a greater parity between the strategic and technical axes of aerospace and defense enterprises. Determining *all* the attributes constituting stakeholder value—not just the technical attributes—is a critical enabler of attaining this parity. Delivering that *total* value entails a process with a greater amplitude than classical Product Realization: it requires a suitably capable *Solution Realization* process.

4.1.2 Process Multi-Modality

One major consequence of the drastic diminishment of the aerospace and defense business base is that far fewer business opportunities exist in traditional markets, making capture of these a matter of survival and pursuit of new opportunities outside of these markets essential for sustainable growth. Two readily accessible opportunities for growth are commercial applications of advanced technology and complete logistical support of their own systems and those of others. The former represents a means to exploit their vigorous innovation streams; the latter a means to exploit their prodigious manufacturing capabilities. The strategic and technical ‘spin-back’ potentials of both would convey significant benefits to the core business as well.

However, these and any other significant opportunities for aerospace and defense enterprises are situated in business environments that differ substantially from their traditional venues. Dominated by market-driven commercially-oriented business models, the structures and modalities of these environments are essentially foreign to classical defense contractor enterprises. That is, the requirements-driven engineering-oriented business model of classical aerospace and defense enterprises reflects the modalities of their core businesses. Their operating systems—specifically their constituent enterprise processes—evolved within these lines and would not, accordingly enable defense contractors to function competitively in purely commercial environments.

A diversification technique called *divisionalization* is frequently employed to effect entry into new markets while protecting existing positions in traditional core businesses. This typically involves creating a new semi-autonomous ‘strategic business unit’ (SBU) in one form or another. Divisionalization can constitute an optimal solution to a variety of strategic and operational problems, and is, in some cases, the only possible solution given regulatory constraints. Nevertheless, from a formal enterprise engineering perspective, divisionalization is *articulation*, to invoke the relation of that name defined in Section 3, and this solution to business model heterogeneity is *not* the only one available. Subsumption—i.e., enterprise process generalization or *multi-modality*—is another solution. Cogently designed enterprise processes capable of executing multiple business models (or substantively divergent variants of a single model) can, in the right circumstances, yield the same performance and effectiveness gains as articulating or creating distinct processes, without the consequential duplication, integration problems, and loss of commonality.

4.1.3 Action Multilaterality

There are two fundamental ways an enterprise can execute a process. The first is unilaterally—the enterprise acts alone or at least is the dominant execution and governance agency of the process. The second is multilaterally—the enterprise acts in concert with others. Unilateral action is effi-

cient but requires total command of all the resources and capabilities to execute and govern a process. Multilateral action is less efficient because it requires coordination. However multilateral action does not require total command of all the resources and capabilities to execute a process; it is a strategy to accomplish stakeholder purposes with less.

The intensely competitive and global business environment places severe constraints on the resources available to an enterprise to achieve and sustain success. Protecting core market share requires sustaining an adequate business capture rate. Penetrating new markets requires balancing investment with risk. The capability to act in multilateral partnerships with customers, suppliers, and competitors is the most effective means available to achieve both of these objectives in the current business environment.

4.1.4 Totality and Autonomism

Toyota launched their “Toyota Production System” (TPS) initiative in the 1950’s to integrate processes, people skills, technology and information across a core element of their enterprise—the factory floor. Their purpose was to create a *product realization system*; an operating system focused on manufacturing and assembly of Toyota products. More recently, many other organizations have pursued similar but broader initiatives to design and deploy enterprise-wide operating systems. Notable private sector examples are those of Chrysler and Alcoa; the Acquisition Reform initiative of the U.S. Department of Defense is an illustrative public sector example.

These recent initiatives are incremental steps towards satisfying the need for all-inclusive high-performance enterprise operating systems. Each has yielded compelling tactical benefits and clear strategic advantages. Collectively they also exemplify two limitations, albeit to varying degrees. The first is *partiality*. No existing operating system in fact fully encompasses its respective enterprise; each is *incomplete* to some extent. All tend to reflect an emphasis on one or more areas of application over others; each is *suboptimal* in some respect. Thus despite the very real value delivery improvements these initiatives demonstrate, the enterprise operating systems produced by them are, nevertheless, partial rather than *total* solutions.

Existing operating systems require exceptional expertise to sustain and prodigious efforts and expense to modify them—they are *fragile*. The fluidity of current business environments coupled with the fragility of existing operating systems entails more effective mechanisms for change. Enterprise operating systems are products of a process we call Enterprise Realization, and changing them involves an iterative execution of this process, *in situ*. This process is outside the scope of existing operating system capabilities to execute directly; that is, Enterprise Realization is not a process of existing enterprise operating systems. Enhancing the effectiveness of enterprise change entails embedding this process into operating systems—i.e., devolving the task to changing an operating system to the operating system itself.

4.2 Enterprise Operating System Requirements

For presentational purposes we have divided the requirements entailed by the four enterprise operating system needs outlined in paragraph 4.1 into three distinct groups—methodological requirements, metastructures, and design requirements. The latter have been further subdivided into mission definition and enterprise process requirements.

The requirements presented here reflect our focus on systematic issues during Mereos Phase III and our formalist role in LEAP and the subsequent LM Aero consolidation program. These should not, therefore be construed in any way as constituting a complete requirements specification for an enterprise operating system satisfying the needs articulated above. As we stated earlier, the NAE operating system design project is a work in progress continuing under other auspices. What follows describes requirements that were defined by the end of the MEREOS program.

4.2.1 Methodological Requirements

The shortcomings exemplified by existing operating systems together with the needs identified in paragraph 4.1 are conjunctively symptomatic of a *methodological gap*. All prior operating system realization initiatives have been based on relatively informal processes that rely heavily on organizational experience and best practices in combination with methods and technology such as those mentioned above. But the need to increase scope and effectiveness of these initiatives will generate concomitant stresses on the realization processes—particularly on underlying methods. Existing methods are simply incapable of producing *total* enterprise operating system solutions at the requisite levels of scale, performance, and sustainability; the incompleteness, suboptimality, and fragility of systems produced by these methods are the results. This methodological gap is reminiscent of the one that afflicted the early years of the space program. The family of methods now collectively known as Systems Engineering were developed to fill that gap. An analogous discipline, which we call enterprise engineering, is emerging to fill this one.

Enterprise engineering is an embryonic interdisciplinary field concerned with design, deployment, and sustainment of enterprise operating systems. Enterprise engineering accordingly comprises formal processes, methods, and techniques for producing and sustaining these systems.

Enterprise engineering is a new synthesis of several existing disciplines. The dominant two of these are systematics and systems engineering. At first glance, enterprise engineering bears a strong resemblance to the latter. Like that discipline, it is *formal*; many of its central methods *are* (suitably modified) systems engineering methods; and, it has also emerged in response to a need arising from unique historical circumstances. However, one can be easily misled by this resemblance, as is illustrated quite nicely by a quote from a recent article in Aviation Week:

“This industry [A&D] invented systems engineering, but traditionally it has been applied to only half of the business. If companies applied the same expertise to *work*, products would cost a lot less, quality would be substantially better, and products would reach the consumer much faster.” [6] (Italics ours)

The fundamental idea expressed above is of course entirely sound. Who could rationally argue against the benefits of bringing systems engineering rigor, traceability, and balance to the task of designing ‘work’—i.e., enterprise processes? However the implicit premise that ‘work’ is the same kind of thing as ‘product,’ and, therefore, that designing either of these is *just* a systems engineering job is naive. It is also *false*. The LM Aero enterprise operating system (to invoke an example) is *not* the same kind of thing as the air vehicle systems it produces. Specifically, it is more complex, more abstract than any product system it produces, and it comprises intentional elements in ways distinct from those of any other existing system.

1. Complexity

One signature characteristic that distinguishes enterprises and their operating systems from any other artifact producible by current technics is that they are *more complex* than the solutions they produce.⁴⁷ And, they are more complex in three distinct ways: they are taxonomically, integrally, and frequently numerically more complex than any of their offerings. Consider LM Aero and its operating system on the one hand and the F-22 on the other as examples. Clearly there is a much larger range of taxonomic variation—differences in

kind—among the elements constituting LM Aero than those constituting the F-22. Moreover, the degree of *intricacy*—the number of interfaces between and relations among elements—of LM Aero far surpasses those between elements of the F-22. And finally, at least in this specific example, the sheer number of LM Aero elements far exceeds that of the F-22.

2. Abstraction

A second characteristic distinguishing enterprises and their operating systems from any other artifact producible by current technics is that they are *more abstract* than the solutions they produce.⁴⁸ And, they are more abstract in two distinct ways: they are quantitatively *and* qualitatively more abstract than any of their offerings. Consider again LM Aero and the F-22. Both the functional and physical architectures of LM Aero encompass a far larger quantity of abstract entities than do those of the F-22. Moreover, significant numbers of these abstract elements are of a much higher rank, or ‘distance’ from the concrete world, than are any of those constituting the functional and physical architectures of the F-22.

3. Intentionality

A third and perhaps the most far-reaching characteristic differentiating enterprises and their operating systems from any other artifact producible by current technics is that enterprises are literally *intentional* or conscious *beings*, and enterprise operating systems comprise intentional beings to a far greater extent and depend on them to a greater degree than any enterprise offering.⁴⁹ The vast majority of extant legal systems treat businesses and as well as types of enterprises as *individuals*—as members of their respective societies, and like any other individual, these legal systems entitle them with certain rights and charge them with certain responsibilities within those societies. Enterprises such as LM Aero, like any other intentional beings, sustain intentional states, are autopoietic, autonomous, and auto-ontogenetic. No artifact producible by current technics exemplifies any one of these characters, let alone all three of them conjunctively.

These quantitative and qualitative differences in complexity, abstraction, and intentional constitution distinguishing enterprises and their operating systems from the products they produce have far-reaching methodological consequences. The constellation of issues pertaining to scale, intricacy, and formal and intentional structures as subjects in their own rights take on a much larger and much more critical role in an enterprise engineering context than they do in classical systems engineering. Simplistically speaking, airplanes don’t design, build, and support airplanes, nor do they develop and execute strategies, nor do they upgrade themselves—but enterprise operating systems are intentional constructs that must do all of these things and more. While systems engineering methods are capable of meeting the difficulties arising from greater complexity, albeit not without some degree of stress, they are not capable of satisfying other unique methodological requirements for successful operating systems design.

Thus enterprise engineering cannot be just a variant or application of systems engineering. It must instead constitute a *new synthesis* disciplines comprising the requisite unifying principles and techniques as an integral gestalt. These are:

1. Systematics: the science of diversity [18.1], concerned with the formal structures of variation, taxonomy, and classification, which provides the integrating metastructures and con-

⁴⁷ Of course one of the principal aims of advancing the engineering and manufacturing arts—specifically automation—is to ‘invert’ this, thereby enabling simple enterprises to render extraordinarily complex solutions. The one-person cnc machining job shop is a good example of this.

⁴⁸ One can characterize many of the advances in the engineering and manufacturing arts as aiming to invert this also—to enable creation of increasingly more abstract artifacts, thereby offering greater capability with commensurate decreases in requisite instrumentality, or entirely new, previously unrealizable capabilities. Software and electronic computing machines (versus mechanical ones) are examples of the former; genetic engineering is an example of the latter.

⁴⁹ The enterprise of so-called Artificial Intelligence seeks to eliminate this distinction by creating conscious artifacts, but despite the claims of some of its market-driven proponents, this has yet to be achieved.

ceptual framework for enterprise engineering.

2. Systems Engineering: the discipline concerned with design of life-cycle balanced systems, which performs the analogous role in enterprise engineering of enterprise operating systems design.
3. Strategic Management: the discipline concerned with action program definition and governance, which in enterprise engineering provides, via its methods, techniques for stakeholder intent definition and enterprise capability analysis, and because of its subject domain, a source for strategy process design.
4. Informatics: the discipline concerned with design and development of effective (computable) representation systems, which is used pervasively in enterprise engineering both as a basis for information systems design and for enterprise engineering automation.
5. Ontology: the “science of Being,” the discipline that, in conjunction with a suitably informed phenomenology, is concerned with the forms of form and intentionality. Its role in enterprise engineering is analogous to the role of mathematics in the natural sciences; it provides the methods and techniques for the analysis of abstractions—i.e., pure formal structure.

Thus enterprise engineering is a *new* discipline, albeit in a synthetic sense rather than an analytic one, and it is, accordingly, something over and above the mere sum of its elements. There are nevertheless functional and historical analogies between enterprise engineering and systems engineering that, when articulated, serve to illuminate the crucial role enterprise engineering will play in advancing business design, execution and management. Unfortunately this topic is outside the scope of our discussion here. What we can say is that systems engineering evolved to address a methodological gap, which was the root cause of several *technical* failures suffered by the American space program in its early days. Enterprise engineering is emerging to address an analogous but distinct methodological gap, which is the root cause of the shortcomings of current enterprise operating systems. These limit operating system capabilities to consistently deliver superior value and effectively respond to change—the two essential ingredients of *strategic* success.

4.2.2 Metastructure

		TYPE	RELATION OF	RELATION BETWEEN↔	EXAMPLES	
CORE	1	COMPOSITION	Containment	Entities↔Elements	ENTERPRISES↔{SUBSIDIARIES,DIVISIONS,ORGANIZATIONS}	
	2	CONSTITUTION	Materiality	Entities↔Materials	OPSYSTEMS↔{FISCAL ASSETS,INFORMATION,ORGS}	
	3	INHERENCE	Characterization	Entities↔Attributes	OPSYSTEMS↔{CAPABILITY,RELIABILITY,EFFICIENCY}	
	4	QUALIFICATION	Conditionality	Relations↔Antecedents	{STAKEHOLDER PURPOSE ₁ ,... _n }↔BALANCED SOLUTIONS	
	5	QUANTIFICATION	Multeity	Relata↔Quantities	FISCAL METRICS↔VALUES; PRODUCTS↔MARKET SHARES	
ADJUNCT	INTRINSIC	6	EQUIVALENCE	Substitutionality	Relata↔Substituends	MIL-Q-9858↔ISO9000; SUPPLIER ₁ ↔SUPPLIER _n
		7	ALTERNATION	Optionality	Relata↔Alternatives	MATERIEL↔{MAKE,BUY}; STRAT PGM↔{CONTINGENCY ₁ ,... _n }
		8	VARIATION	Diversity	Baselines↔Variants	PRODUCT REALZTN↔{DOD VARIANT,COMRCL VARIANT}
		9	ORDER	Positionality	Entities↔Positions	ACTION ₁ ↔ACTION ₂ ; OBJECTIVES↔PRIORITIES
		10	TRANSFORMATION	Development	Entities↔States	COLD WAR A&D↔POST-CW A&D; EPA REGS v1999↔2000
	EXTRINSIC	11	ARTICULATION	Separation	Entities↔Realizations	ENTERPRISES↔{STRATEGIC BUSINESS UNIT ₁ ,... _n }
		12	FACTORIZATION	Abstraction		{TASK ₁ ^{typeA} ,...,TASK _n ^{typeA} }↔FUNCTIONAL ORG
		13	CONSOLIDATION	Integration		{TASK ₁ ^{typeX} ,...,TASK _n ^{typeY} }↔PROCESS ORG
	INTERTYPE	14	INVOLVEMENT	Participation	Continuants↔Occurs	OPSYSTEMS↔ENT PROCS; FISCAL ATTRS↔AUDIT PROCS
		15	CAPACITATION	Provisionment	Involvements↔Tools;Agencies	OPSYSTEMS↔{FORMAL METHODS,INTEL,HUMAN ASSETS}
16		REPRESENTATION	Objectification	Entities↔Representations	INTENTS↔MISSION STATEMNTS; RULES↔POLICY STMTS	
17		DESIGNATION	Nominalization	Entities↔Designators	ENTERPRISES↔LOGOS; PRODUCTS↔BRAND NAMES	
TAXONOMIC	18	CORRELATION	Complementation	Taxa↔Complements	ENTERPRISES↔{ARCHITECTURES,OPSYSTEMS,PRODUCTS}	
	19	DELIMITATION	Differentiation	Taxa↔Characters	PROCESSES↔MOES; PRODUCTS↔COST/QUAL/PERFMNCE	
	20	IDENTIFICATION	Membership	Taxa↔Keys	INDUSTRIES↔SIGNATURES; AGENCIES↔BEHAVIORS	
	21	CLASSIFICATION	Rank	Taxa↔Categories	POPULATIONS↔{SUBSPECIES,SPECIES,GENUS,...,DOMAIN}	
	22	DEVIATION	Irregularity	Delimitations↔Anomalies	PRODUCTS↔DEFECTS; PROCESSES↔CAPABILITY GAPS	

Table 7. Enterprise Operating System Definition Metastructures

Enterprise operating systems are products of the Enterprise Realization process, a *variant* of the Product Realization process. We pointed out the analogies between these two processes and their respective products elsewhere, and will not repeat that discussion here. What we have *not* previously defined are the structural *requirements* for producing adequate *definitions* of enterprise operating systems, their elements, and the relations between these.

To gain an insight into the purpose of these requirements in the context of operating system definitions, consider the analogous relationship between the structures defined by ANSI Y 14.5 (“Geometric Dimensioning and Tolerancing”) and feature definitions of mechanical product system parts. This specification describes requirements for defining *maximally interchangeable* parts. It accomplishes this by prescribing how both the positional and geometric characteristics are to be defined, and how allowable positional and geometric tolerances for these characteristics are to be defined. The metastructures represent an analogous mechanism for defining *maximally rigorous and complete* operating system elements. This is accomplished by prescribing *what* structures such definitions must define, *how* these are to be specified, *conditions* for their use, and finally, the mechanism for defining *new* derivative structures from the fundamental ones. Thus the 22 specific relation types depicted above in Table 7 collectively constitute the necessary and sufficient metastructures for defining enterprise operating systems and their elements satisfying the four enterprise operating system needs previously outlined in paragraph 4.1. That is, definitions of any enterprise operating system solutions to those needs must be explicitly framed in terms of these 22 structures.

The reader will no doubt note that the first 17 of these structures are the same as those enumerated in Table 1 on page 37 in Section 3 of this document. The only differences between the two tables are the examples presented to illustrate their applications. This coincidence is not accidental:

enterprise operating systems *are* products in every sense of that term, although as we pointed out earlier, they differ from more familiar product systems in complexity, abstractness, and intentional constitution. Because operating systems are products, there is no reason to suppose that the metaschematic requirements for their *definitions* would differ from those for any other complex product system definition, excepting of course those requirements deriving from the aforementioned differences individuating operating systems from other types of systems. And, in fact, almost all of these metaschematic requirements *are* the same. Hence most operating system metastructure requirements have already been defined in paragraph 3.2.1 of Section 3; these are incorporated herein by reference.

The remaining 5 relation types listed in Table 7 collectively constitute the metataxonomic structures required to define enterprise operating system elements and attributes that are not found in typical product systems—that is, those which are unique to these types of systems. These structures are extraordinarily abstract, and would require an extensive introductory presentation followed by a very lengthy discussion to adequately describe them. As we do not explicitly invoke or refer to any of these structures in this document, we elected to not to define these here.⁵⁰

4.2.3 Mission Definition

Mission statements articulate the primary and invariant purposes of enterprises—they define *what* is to be accomplished, independently of circumstantial specifics. Strategies delineate programs for achieving these and other consonant but more particular aims—they define *how* missions are to be achieved. Enterprise operating systems are instrumentalities for purposeful collective action—they comprise the *mechanisms* for executing these programs. Mission statements are, accordingly, critical for both rational strategy development and systematic operating system design. They constitute the ultimate grounds for action, the ultimate criteria for evaluating results, and the ultimate source of design requirements.

Mission definition requirements in an enterprise engineering context differ from those in familiar strategic management contexts. In the latter they articulate primary organizational imperatives to a human audience, and should, accordingly, exemplify the attributes appropriate to that purpose. From an enterprise engineering perspective, mission statements are enterprise operating system *needs statements*, and must, therefore, be designed with *that* purpose in mind. The following quotation below will help illustrate this point more clearly.

“The ‘mission’ of a company is an important element in establishing the strategy of the organization. Establishing the mission itself is usually a difficult and demanding task. Top management tends to agonize for long periods of time over the development of a mission statement: the process involves negotiation and compromise, but is usually leadership led—and depends upon critical input from the ceo. Surprisingly, perhaps, despite all the effort expended, many mission statements tend to seem full of platitudes and motherhood statements.

...Mission statements need to be communicated throughout the organization....

Good mission statements tend to be simple and easy to understand at all levels of the organization.... For example, in the US General Electric Company, the mission for each business is to ‘be number one or two in the world or sell it, close it, or fix it.’ Such a statement is readily understood and memorable.” [4.1]

Based on these statements, we can frame a thesis defining the requisite attribute of a mission definition in a strategic management context:

Good mission statements consist in *transparent* expressions of strategic intent.

By “transparent”, we mean *symbolic*. That is, statements which are sufficiently *evocative*—allusive and mnemonic—to engender universal understanding of the aims these statements express, and to pro-

⁵⁰ Excepting CORRELATION, these structures are the principal structures underpinning the discipline of systematics and are, accordingly, described by that body of literature. Although now out of print and therefore somewhat difficult to find, we would highly recommend that anyone interested in this topic acquire a copy of *Principles of Systematic Zoology* [18] and study it.

mote operable concurrence among various interests concerning those aims. In a word, statements that impel *focus*.

Although memorability and evocativeness are useful attributes in an enterprise engineering context, something else is required, which we state as a another thesis:

Good mission statements consist in *effective* expressions of strategic intent.

By “effective,” we mean *actionable*. That is, statements which are sufficiently *definitive*—specific and precise—to enable systematic development of operating systems capable of executing the action programs required to achieve the aims these statements express, and to facilitate cogent selection of measures of effectiveness for evaluating both designs for these systems and outcomes of their actions. In a nutshell, statements that sustain *operational traceability*.

Producing an effective mission definition *instance* presupposes a suitable definition of the MISSION TAXON itself. Figure 18 graphically depicts the definition of this taxon we developed during Mereos Phase III, and a brief description of its elements follows.

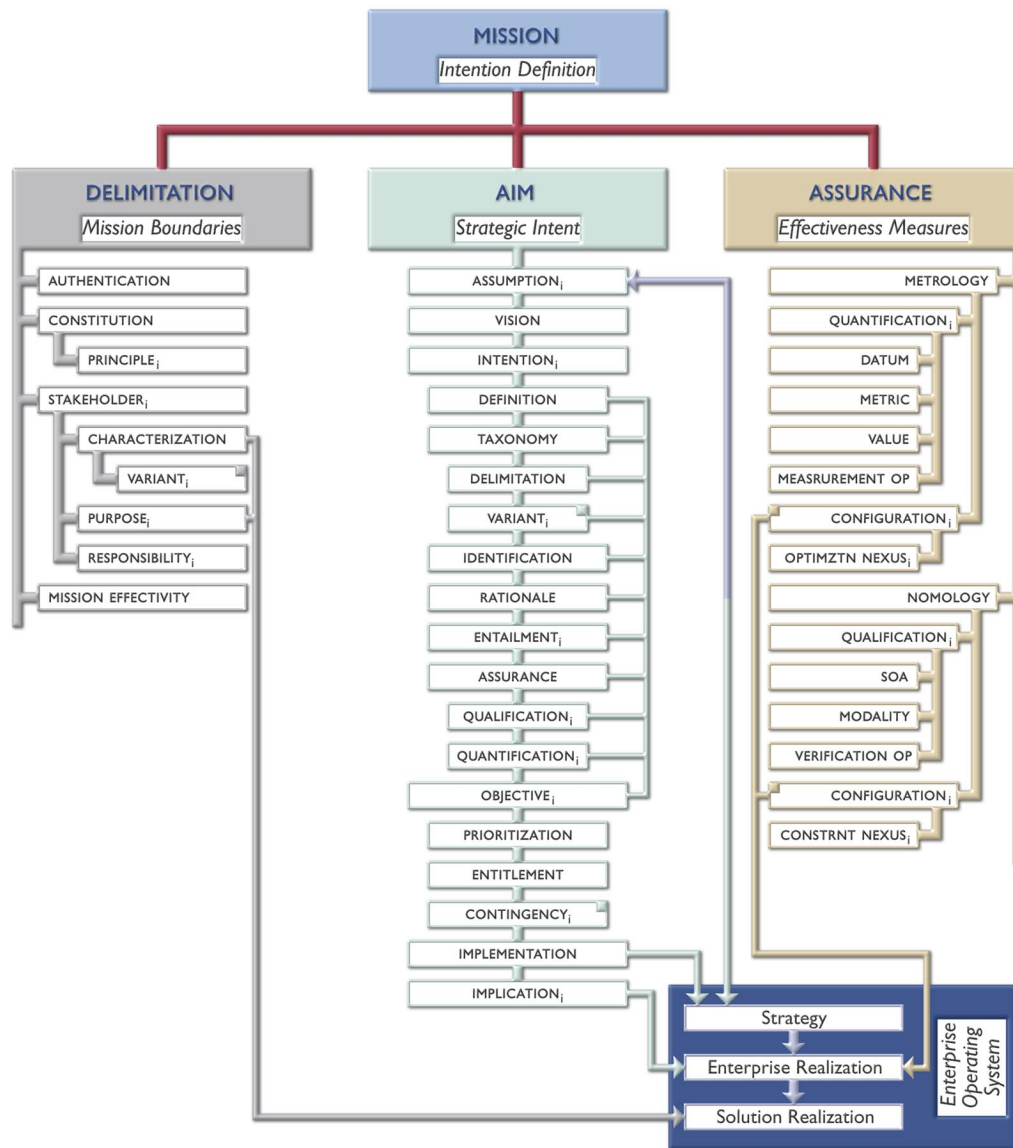


Figure 18. Mission Definition Elements

I DELIMITATION

The four major elements constituting the DELIMITATION element of the MISSION taxon collectively define the situating environment of its strategic intentions including the agencies that sustain those intentions, thereby determining the boundaries of permissible action for achieving that mission.

AUTHENTICATION

The authentication element of the MISSION taxon identifies the agency authorizing the enterprise to undertake that mission.

CONSTITUTION

Not all actions in pursuit of a mission are permissible. There are constraints on what actions the enterprise can take to achieve its mission. These constraints are expressed as 'value' statements or *fundamental tenets*, and are embodied by core values that govern the actions of the enterprise and its stakeholders. The constitution element of the MISSION taxon defines these fundamental tenets that guide the actions of the stakeholders in the enterprise. These tenets represent the sources from which the nomological elements of the ASSURANCE element are derived.

STAKEHOLDERS

The stakeholders element of the MISSION taxon identifies those agencies possessing interests in the enterprise. These agencies comprise individuals, corporate entities, governments, and societies whose strategic purposes actually or potentially entail the enterprise as an instrumentality for accomplishing those purposes. Business enterprises typically comprise 9 primary stakeholder groups, as enumerated below.

1. Customers

Customers are agencies that receive goods and services from an enterprise in return for an agreed upon valuation in some medium of exchange.

2. Employees

Employees are individuals that receive regular compensation from an enterprise in exchange for regular performance of assigned tasks. MANAGEMENT is a variant of EMPLOYEE.

3. Superiors

Superiors are agencies that authoritatively define superordinate intent and effectiveness measures to which all enterprise intent and assurance is subordinate, and which cede entitlements to provision an enterprise to execute that intent.

4. Stockholders

Stockholders are agencies that own or hold one or more shares in an enterprise.

5. Suppliers

Suppliers are agencies that furnish goods and services to an enterprise in return for an agreed upon valuation in some medium of exchange.

6. Unions

Unions are agencies consisting of employees of one or more agencies that share common interests and act collectively to influence employment policies and practices.

7. Peers

Peers are agencies satisfying one of the following two conditions.

- 0.1. A peer is an agency that shares an enterprise's superiors, is of equal or analogous rank to an enterprise in superordinate context, and thus is an executor of the same superordinate intent as that enterprise.

- 0.2. A peer is an agency that does not share an enterprise's superiors, shares at least one strategic or tactical intent with that enterprise, and has explicitly agreed to pool resources, coordinate actions, and share the products and effects resulting from those actions in pursuit of that intent. These latter are partners.

8. Communities

Communities are agencies consisting of individuals acknowledging a unity of purpose, common interests, or common culture.

9. Competitors

Competitors are agencies offering similar or equivalent products and services in the same markets as those offered by an enterprise, who are engaged in executing similar or identical intent, and who are not peers or superiors of that enterprise.

EFFECTIVITY

The effectivity element of the MISSION taxon delineates the conditions under which a mission is operant for an enterprise.

■ MISSION INTENT

The central elements of the MISSION taxon are statements of strategic intent. Definitive characterizations of strategic intents consists in rigorous definitions of their elements and the elements of their situating contexts, produced via the Intent Formalization process described in paragraph 4.2.4.2 below.

■ Mission Assurance

The two major elements constituting the assurance of the MISSION taxon delineate measures for evaluating the effectiveness of action programs undertaken to achieve the aims of that mission. A brief discussion of these elements is presented in paragraph 4.2.4.3 below.

4.2.4 Enterprise Processes

One of our efforts consisted of requirements definition for and schematic designs of enterprise operating system processes. Our specific tasking was to provide systematic answers to the following questions.

1. What is an enterprise process?
2. What are the major enterprise processes?
3. Which of these are the most critical to overall strategic success?
4. What are appropriate performance measures for each of the enterprise processes?

We produced three principal products in response to this tasking. The first was a formal characterization of all principal enterprise operating system processes. A summary of this is presented in the next paragraph. The second was a detailed outline of the INTENT FORMALIZATION process, which is one of the three critical enterprise processes from both a strategic management and an operating system perspective. This is presented in paragraph 4.2.4.2. The third was preliminary work on a taxonomy of Measures Of Effectiveness (MOEs). The specific issues we addressed on that topic are summarized in paragraph 4.2.4.3.

4.2.4.1 Enterprise Process Characterizations

■ Definition

The following is an abstract of an exhaustive formal delimitation of the taxon ENTERPRISE PROCESS which we produced in response to the question:

What is an enterprise process?

This result enabled practitioners to systematically determine whether a given process under consideration is, in fact, an enterprise process, a subsidiary process, a function, or an action, or an enterprise-specific implementation of one or more of these. Such determinations are critical steps in operating system design.

DEFINITION: ENTERPRISE

Enterprises are agencies for *rendering solutions* to requirements that ultimately originate from stakeholder purposes. Accordingly, enterprises are *intermediate* instrumentalities for accomplishing those aims—the solutions they produce *provision* stakeholder operations for achieving them. We can readily infer three formally useful facts from this informal characterization by structuring it in terms of the INSTRUMENTALIZATION relation scheme.⁵¹

1. As enterprises are implementation agencies, they necessarily stand in AGENCY attribute positions of INSTRUMENTALIZATION relations. Thus, for a given enterprise x :

$\langle \text{Instrumentalization}, \langle \text{Involvement}, \text{Enterprise}_x, \text{Product} \rangle \rangle$.

2. From the above it easily follows that solutions rendered by enterprises necessarily stand in PRODUCT attribute positions of these relations. Thus again for a given enterprise x :

$\langle \text{Instrumentalization}, \langle \text{Involvement}, \text{Enterprise}_x, \langle \text{Solution}_1, \dots, \text{Solution}_i, \dots, \text{Solution}_n \rangle \rangle \rangle$.

3. Purpose/operation *pairs* constituting relations between stakeholder aims and accomplishment methods are the sources of solution requirements. Accordingly these necessarily stand in INVOLVEMENT attribute positions:

$\langle \text{Instrumentalization}, \langle \langle \text{purp}_1 \rangle [\text{op}_1, \dots, \text{purp}_i] [\text{op}_i, \dots, \text{purp}_n] [\text{op}_n] \rangle, \text{Enterprise}_x, \langle \text{Solution}_1 \dots \text{Solution}_n \rangle \rangle$.

Thus in the context of the current question, *an enterprise* is an entity instantiating the AGENCY attribute of at least one INSTRUMENTALIZATION instance:

$\langle \text{Instrumentalization}, \langle \langle \text{purp}_1 \dots \text{purp}_n \rangle [\text{op}_1 \dots \text{op}_n] \rangle, \text{Enterprise}_x, \langle \text{Solution}_1 \dots \text{Solution}_n \rangle \rangle$

DEFINITION: PROCESS

Processes are regular and definite successions of interrelated activities that yield consistent and determinate results. Enterprise processes are customarily characterized in the management literature as specific variants exemplifying two additional characteristics. The first is *applicability*—that their outputs necessarily constitute ‘value’ to the ‘customers’ of the process. The second is *extensive totality* or *completeness*—that they encompass an entire ‘value chain’ relative to the ‘customers’ of the process. This attribute is why many call enterprise processes ‘horizontal;’ or ‘end-to-end’ processes. We will explicate completeness now; applicability will follow that.

Complete processes are *ontogenies*—courses of activities constituting entire entity life cycles from beginning to end. We can usefully recast this informal characterization in terms of the REALIZATION relation scheme,⁵² thereby establishing the formal fact that complete processes necessarily stand in the OCCURRENT attribute positions of REALIZATION relations. Thus for a specific complete process CP encompassing the life cycle of a given entity x we obtain:

$\langle \text{Realization}, \langle \text{CP}_x, \text{Inception}_x, \text{Continuation}_x, \text{Termination}_x \rangle \rangle$

There are three positional variants of complete processes of interest to us here: *primary*, *secondary*, and *tertiary*. These variants are defined in terms of *inverse reflective* REALIZATION order: 1st-order REALIZATION instances are complete tertiary processes; 2nd-order REALIZATION instances are complete secondary processes; and, 3rd-order are of course primary. Secondary or *major subprocesses* of complete processes are *phases* of ontogenies—successions of actions or subprocesses constituting all three subsidiary subphases of one of three specific stages of entity life cycles. That is, secondary processes also necessarily stand in the OCCURRENT attribute positions of REALIZATION relations. How-

⁵¹ INSTRUMENTALIZATION is a variant of the CAPACITATION relation type defined in paragraph 3.2.5.2 of Section 3.

⁵² REALIZATION is the occurrent variant of the CORRELATION relation type.

ever, as these are *subprocesses* of complete processes, instances of these relations are necessarily situated in at least one containing *REALIZATION* instance. Thus for a specific secondary process *SP* encompassing the *INCEPTION* phase in the life cycle of entity *x* we obtain:

$$\langle \text{Realization}, \langle \text{CP}_x, \langle \text{Realization}, \langle \text{SP}_x, \text{Incept}^{\text{INCEPTION}}_x, \text{Cont}^{\text{Continuation}}_x, \text{Term}^{\text{Termination}}_x \rangle \rangle, \text{Continuation}_x, \text{Termination}_x \rangle \rangle$$

where for $\text{Incept}^{\text{INCEPTION}}_x$ read “*INCEPTION phase of INCEPTION phase of x*”, and so forth.

Tertiary or subprocesses of complete process subprocesses are *REALIZATION* “leaves” relative to a particular ontogeny. That is, they comprise the ‘lowest-level’ 1st-order variants of complete processes; all elements of complete tertiary processes are either partial processes, actions, or functions. Tertiaries are *phases of phases* of ontogenies and, as one can easily infer, they are successions of actions or subprocesses constituting all three atomic phases of a subsidiary subphase of a specific entity life cycle stage.

DEFINITION: ENTERPRISE PROCESS

In light of the preceding definitions, enterprise processes are clearly *REALIZATION processes of INSTRUMENTALIZATION relation elements*, satisfying exactly one of the following four conditions:

1. The inputs to the process are requirements originating from at least one stakeholder purpose, and the results of that process *are* solutions to those requirements. That is, the process is a *solution realization* process executed by the particular enterprise in question.
2. The inputs to the process are one or more stakeholder purposes, and the outputs of that process are input requirements to a solution realization process *and* directives governing execution of all enterprise processes, including that process itself. That is, the process is *strategy realization* (i.e., governance; management) process executed by the particular enterprise in question.
3. The inputs to the process are action directives to the particular enterprise in question that are direct outputs of a governance process of that enterprise, and the outputs of that process are either (a) inputs to at least one enterprise process, possibly including that process itself, or (b) infrastructure directly enabling execution of at least one enterprise process, possibly including that process itself. That is, the process is an *enterprise realization process* executed by the particular enterprise in question.
4. The process in question is either (a) a complete secondary or (b) a complete tertiary *REALIZATION* process as defined above of an enterprise process satisfying exactly one of the prior three conditions. That is, the process is a major subprocess of an enterprise process or a subprocess of a major subprocess of an enterprise process.

An enterprise process is, accordingly, a process executed by an enterprise that meets at least one of the following criteria:

1. it directly renders a stakeholder solution;
2. it governs the enterprise;
3. it creates, modifies, or sustains the enterprise itself; or,
4. it is a complete secondary or tertiary enterprise process subprocess.

Note also that applicability—that characteristic obtaining when process outputs in fact constitute ‘value’ to the ‘customers’ of the process—is defined as satisfaction of the above criteria.

■ Taxonomy

The following is an overview of a formal taxonomy of enterprise processes which we produced in response to the question:

What are the major enterprise processes?

This result provided practitioners with an enumeration of all first and second-level enterprise pro-

cesses and their primary attributes, as well as third level processes and their attributes. This taxonomy was intended to be used by practitioners for several different purposes, notably to verify completeness of operating system process designs, and to assess alignments of these to an optimal configuration.

It is self-evident from the definition of the ENTERPRISE PROCESS taxon summarized above that there are necessarily *exactly three* principal or ‘top-level’ enterprise processes, as there are no other processes at this rank that satisfy the conditions stipulated by that definition. These are:

1. Strategy Realization
2. Enterprise Realization
3. Solution Realization

These three primary processes are depicted by the rows in Figure 19 below labelled PURPOSE, AGENCY, and ARTIFACT, respectively.

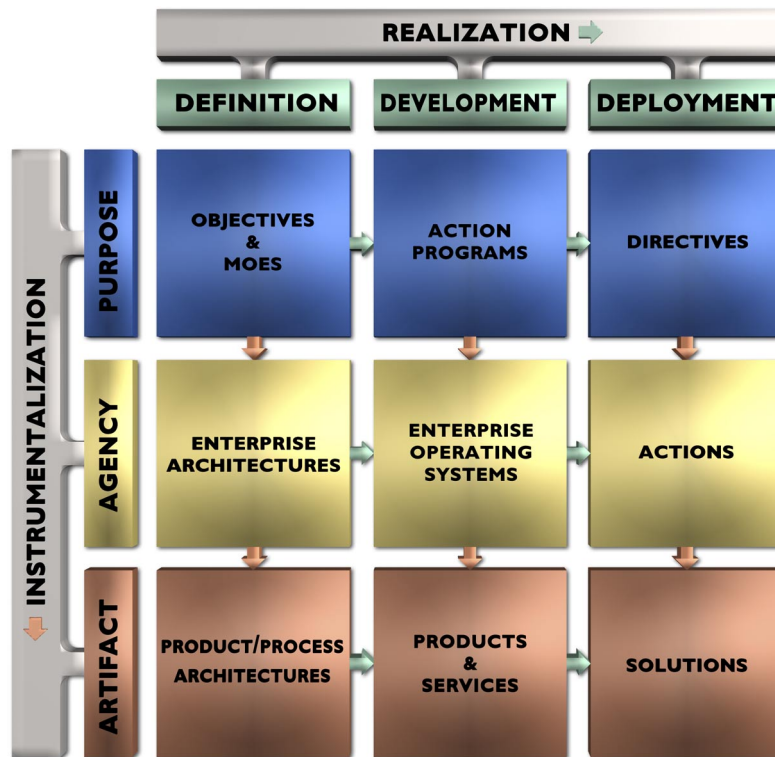


Figure 19. Principal Enterprise Processes

We use term *realization* in a process name to informally signify that the process is life-cycle complete in the sense defined earlier, and to distinguish such processes from those which are not.

The vertical axis of the diagram depicts the INSTRUMENTALIZATION relation between stakeholder purposes the three primary enterprise processes and their products, and is a graphical rendering of our definitional remarks concerning ENTERPRISE above.

The secondary or ‘second-level’ enterprise processes together with their principal products, are depicted by the cells in the diagram. These also constitute the major phases of the primary enterprise processes, the generic names of which are depicted by the columns labels in the diagram. There are *exactly nine* secondary enterprise processes. We will not describe them all here. One of them (INTENT FORMALIZATION, depicted by the upper-left cell) is described in some detail below. The three cells in the bottom row depict the processes commonly called ‘Product Definition,’ ‘Product Delivery,’ and ‘Operations & Support’ in defense contracting circles. These are the major subprocesses of the Solution Realization process. The three cells in the center row depict specific variants

of Solution Realization subprocesses representing those constituting the enterprise operating system life-cycle—i.e., the major subprocesses of the Enterprise Realization process.

The tertiary or ‘third-level’ processes are not shown on this diagram, although one group of them—the subprocesses of Intent Formalization—are depicted in the diagram on page 73. There are *nominally* 27 tertiary processes, all of which satisfy the conditions for being enterprise processes and are, therefore, important operating system processes. Due to similarities among them which enable us to consolidate triples of them into a single process, there are actually only 9 core tertiaries, each encompassing 3 distinct variants. Thus there are, as a formal matter of fact, exactly 21 significant processes constituting an enterprise operating system—9 elemental and 12 enterprise processes. All other processes are accordingly either:

1. Variants of one or more of these processes;
2. Elements of one of these;
3. Variants of elements;
4. Impositions of extrinsic regulatory requirements, or;
5. Unnecessary, and therefore superfluous.

■ Identifications

The following is brief summary of the arguments supporting our identification of Intent Formalization and Solution Realization as the two processes that are most critical for enterprise success, produced in answer to the following question:

What enterprise processes are the most critical to overall strategic success?

This identification enabled practitioners to focus their actions on maximizing the strategic viability of the enterprise.

Enterprises are maximally instrumental (effective) in accomplishing stakeholder purposes when their actions yield *total* solutions to needs and requirements that are *veridical derivatives* of operations executed by stakeholder to accomplish those purposes. A formal definition of TOTAL SOLUTION was one result of our activities in support of LEAP and is not reproduced here, although an extract from that definition is used to illustrate the Explication process in paragraph 4.2.4.2 below. A definition of “veridically derivative” requirement is sketched in that same paragraph.

The primary relationship of interest in enterprise engineering is accordingly that between requirements and solutions—namely, the APPLICATION relation defined in paragraph 3.2.5.2 of Section 3, and depicted by red line in the diagram in Figure 20 below. A principal formal magnitude associated with this relation is *Capability*, defined as *capacity for effective action*, and this is a primary magnitude of interest in enterprise operating system design.

There are exactly two subsidiary positionally distinct attributes of Capability—*Value* and *Quality*. Value is *fitness for possession*. Quality is *fitness for application*. These are the ultimate attributes of interest for optimization in enterprise engineering. Capability is the *only* determinant that matters to stakeholders—it is the quantity of *direct* and *operant* instrumentalization in stakeholder contexts—and Value and Quality are the *only* salient measures of that quantity in those contexts. Thus the central focus of successful enterprise engineering is *Capability Optimization*—maximization of the value and quality of stakeholder solutions.

The two most crucial operating system processes enabling that are Intent Formalization and Solution Realization. Intent Formalization is the process constituting the Definition phase of the Strategy Realization process, depicted by the upper left cell of the diagram in Figure 19. This process is summarized in the next paragraph. Solution Realization is one of the three primary or ‘top-level’ enterprise processes, depicted by the bottom row of that diagram. It is a generalization of Product Realization, designed to address the amplitude and multi-modality needs described in paragraphs 4.1.1 and 4.1.2, respectively. A formal definition of this process was still under development at the end of MEREOS Phase III.

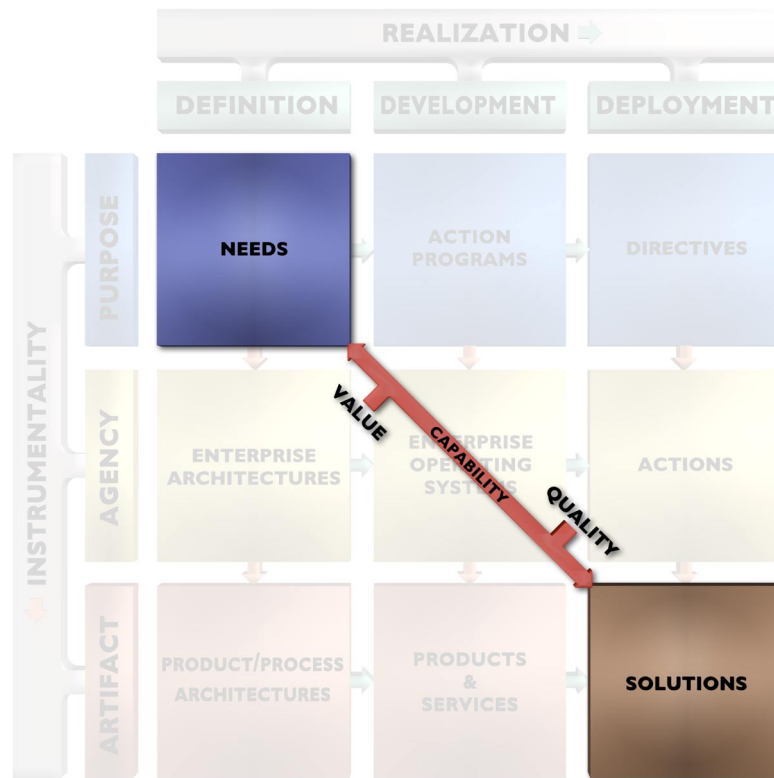


Figure 20. The Capability Relation and Attributes

4.2.4.2 Intent Formalization

The Intent Formalization process is one of the two processes which are critical to the strategic success of an enterprise. This section presents excerpts of a document providing a detailed formal definition of this process. This document provided practitioners with information needed to design an enterprise-specific implementation of this crucial process.

■ Description

Purposes must be explicitly defined to enable actions to be devised and executed to accomplish them. Requirements antecedent to those purposes must be identified to enable the synthesis, development, and use of solutions for implementing actions to accomplish them. Qualification criteria must be explicitly defined to enable governance of these actions, and to enable determination of the value of solutions that is necessary for them to fully provision sufficiently effective actions. Thus *definitization* is crucial to purpose accomplishment.

Some purposes depend on the successful attainment of others in order for them to be accomplished. More precisely, some of the states designated by particular intents articulating these purposes are dependent upon antecedent states. Such states cannot therefore be actualized unless the antecedent states are also actualized. Thus *identification* and *analysis* are crucial to purpose accomplishment.

Purposes typically exemplify tremendous variations in several dimensions. They range from the abstract to concrete; from the unattainable to the trivial; from the short-term to the long-term. Some purposes are identical, some are congruent, some are diametrically opposed. Capabilities and resources are finite. It is, accordingly, almost always impossible for an enterprise to achieve the all of the aims of its stakeholders to the degrees they desire. An achievable and acceptable balance must be struck. Thus *systemization* is crucial to effective and efficient purpose accomplishment.

Intent Formalization is the enterprise process for purpose definitization and systemization. This

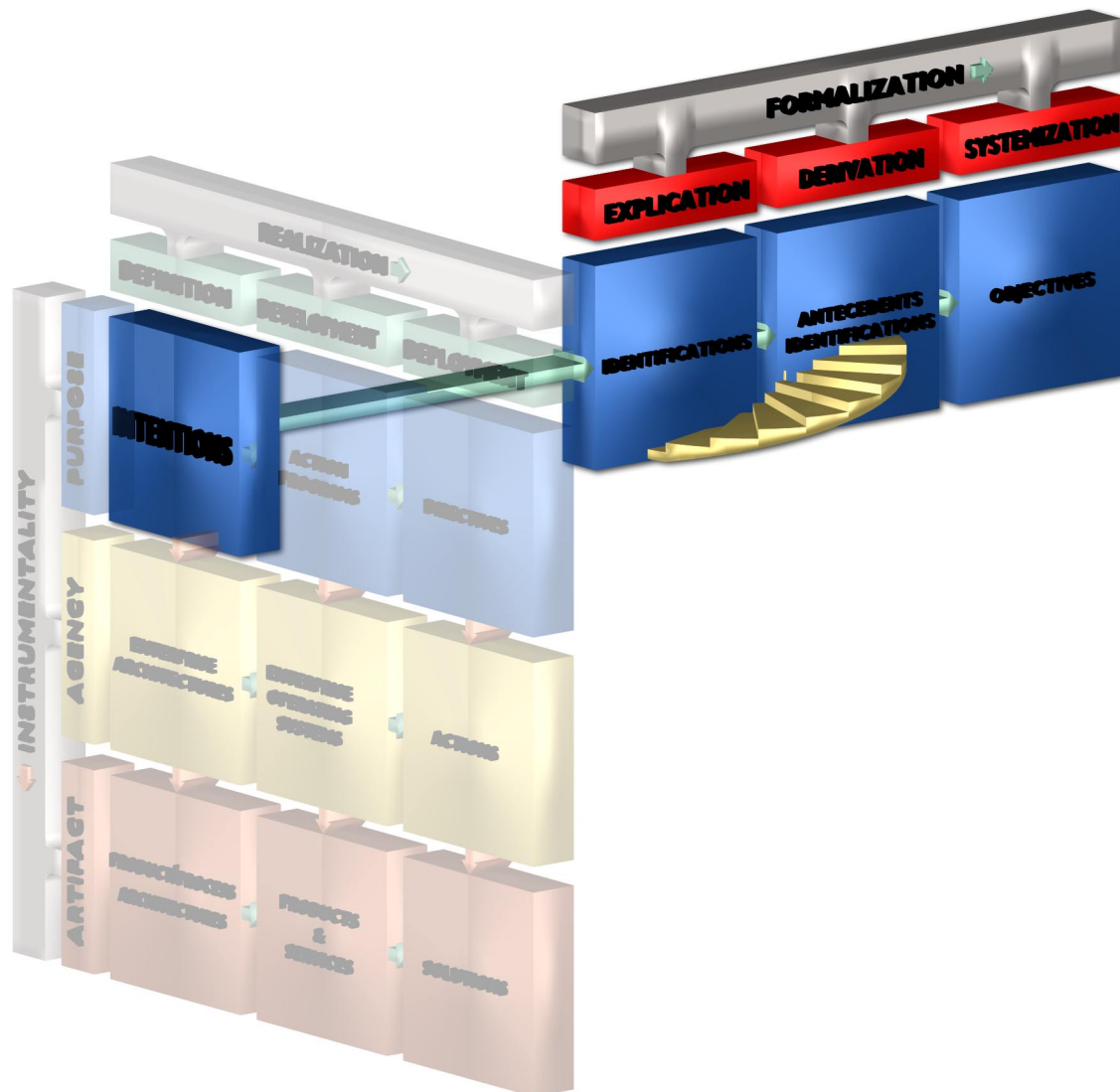


Figure 21. Intent Formalization Elements

process is one of nine secondary enterprise processes and one of the three major subprocesses of the Strategy Realization process.

The primary inputs of Intent Formalization are stakeholder purposes and a formal mission definition for the enterprise.⁵³ Its two primary outputs are *balanced* sets of enterprise objectives representing actionable optimal configurations of stakeholder purposes, and associated measures of effectiveness (MOES) for these. These MOES are accordingly *formal determinations* of requisite *stakeholder value*, as interpreted by enterprise management so as to be consonant with enterprise capabilities, policies, and environmental constraints. Therefore, Intent Formalization is a *generalization* of the process called Customer Value Determination in some management literature.

■ Intent Formalization Subprocesses

There are three subprocesses of Intent Formalization, depicted by the unghosted cells in the diagram in Figure 21 above.

INTENT EXPLICATION

⁵³ A summary of the elements of the Mission Definition structure is presented in paragraph 4.2.3 above.

The term *explication* refers to the process of transforming an informal or incomplete definition into a formal and complete definition—in this context a definition of a stakeholder purpose. Explication is a variant of the formal systematic process called *delimitation*. Intent explication is the process that transforms informal or implicit statements of stakeholder purpose into rigorously formal and systematically complete statements of strategic intent.

The primary inputs to this process are, accordingly, statements of purpose originating from stakeholders themselves and intelligence concerning those purposes. The outputs are fully instantiated and validated instances of the Architecture Element structure, depicted by the drawing in Figure 30 on page 99 in Attachment 2. These comprise fully elaborated and analyzed configurations of the input purposes in that form, including definitive MOEs for accomplishments of the explicated intents. The Architecture Element structure is an adaptation of our metasytematics of TAXON, tailored for this kind of analysis and for downstream enterprise operating system design.

Purposes, strategic intents which articulate them, and objectives representing correlate enterprise aims are all special kinds or variants of *desiderative* intentions. Desiderative intentions are, in turn, special kinds of intentions, all of which of interest to us here are *propositional attitudes* sustained by agencies.

An agency is either a self-aware sentient being that is capable of autonomous action, or an integral collective of these that is capable of united and autonomous action. Examples of the first are human beings; examples of the second are enterprises such as LM Aero.

A propositional attitude is a mental state that consists in a proposition and an *assertoric mode*. An assertoric mode is an intentional ‘orientation’ or kind of attitude of an agency with respect to a proposition. There are several assertoric modes. Four common one are *imperative*, *declarative*, *interrogatory* and *desiderative*. A proposition in imperative assertoric mode is a *directive* that is expressed by a command; one in declarative mode is a *predication* expressed by a statement, one in interrogatory mode is a *query* expressed by a question, and a proposition in desiderative mode is an *aim* expressed by statement of a purpose, intent, or an objective.

A proposition is a variant of phenomenological or cognitive *content* of a mental act that designates or objectifies a *state of affairs* (soa): for our purposes here, a proposition is an soa.⁵⁴ Thus stakeholder purposes, strategic intents, and enterprise objectives are soas representing states that one or more agencies aspire to be actual. Soas were described in paragraph 3.2.1.1, Results Part I beginning on page 28. There we employed soas as a notational device for defining product representation meta-structures. Here their role is objective rather than merely descriptive: the principal entities involved in all three Intent Formalization subprocesses are soas or soa *schemes*—the formal structure of this kind of intentional content—and Intent Explication is, for all practical purposes, *taxonomic delimitation* of the elements of soa schemes representing strategic intents.⁵⁵

As we mentioned earlier in paragraph 4.2.3, the formal counterpart of top-level projective intent defined by the mission definition we helped develop for LMAS during MEREOS Phase III was *Focus On Total Solutions*. We did not complete our explication of this intent before LEAP was overtaken by events; however we have included an extract of this to illustrate what the early ‘descriptive’ stage of

⁵⁴ Those familiar with the content theory of intentionality will no doubt wince at the conflation of content with concept in this definition, not to mention the conflation of the intended soa with its objectification. While we both fully understand and adhere to these distinctions privately and while engaged in our own formal work, they are simply not relevant for our purposes here, and introducing them would make an already difficult and abstract narrative even less accessible than it already is.

⁵⁵ Soa schemes were also described in paragraph 3.2.1.1; as we pointed out there, they are really syntactical shortcuts for DELIMITATION SOAS. Thus Intent Explication really is Delimitation—allowing for our deliberate conflation of intentional content (soa schemes and their elements) with object (taxa and their characters).

this process involves—to convey a “feel” for formal explication.

```
<[ ],<FOCUS,<SOLUTIONS,LMAS>>>>
|
|<TOTAL<INDEXICALIZATION,<FocusSOA,1>>>>
|
```

Focus

Focus is exactly one of the two following types of intentional structure:

1. CONVERGENCE

A focus is a multiplicity of intentions satisfying one of the following two conditions:

- 1.1. Every intention constituting that multiplicity objectifies the same soa as that objectified by the operant intention in the strategic context delimiting that multiplicity of intentions; that is, *all* intentions in that multiplicity are *co-involute* to the operant intention of the delimiting strategic context.

OR

- 1.2. All soas objectified by all intentions constituting that multiplicity are *congruent* to the operant intention in the strategic context delimiting that multiplicity of intentions.

2. INTERDICTION

A focus is a multiplicity of intentions satisfying exactly one of the following two conditions:

- 2.1. Every intention constituting that multiplicity objectifies the same soa; that is, all intentions in that multiplicity are co-involute,

AND

that soa is the *INHIBITION sufficiency condition* for the soas objectified by all operant intentions involved in the strategic context delimiting that multiplicity of intentions which are divergent from the operant intention of that strategic context.

XOR

- 2.2. All soas objectified by all intentions constituting that multiplicity are congruent,

AND

all soa objectified by all intentions constituting that multiplicity collectively constitute the *inhibition sufficiency condition* for the soas objectified by all other operant intentions involved in the strategic context delimiting that multiplicity of intentions which are divergent from the operant intention of that strategic context.

SOLUTION

A solution is a system that is an instrumentality in at least one action, where at least one consequent of that action is an antecedent to accomplishment of at least one strategic purpose.

SYSTEM

A system is a product-process gestalt.

PRODUCT

A product is a continuant that is an output or reduct of at least one occurrent.

In the context of this explication, a product is a continuant that is an output or reduct of an at least one action or at least one process.

PROCESS

A process is a nomologous ontogen.

In the context of this explication, a process is a nomologous ontogen satisfying at least one of the following two conditions:

1. At least one delta of at least one evolute constituting that ontogen is an action.
2. At least one delta of at least one evolute constituting that ontogen is a process satisfying the above condition.

That is, in the context of this explication, a process is a nomologous ontogen in which at least one agency is involved in its realization.

GESTALT

A gestalt is an integral co-concrescent complex.

INSTRUMENTALITY

An instrumentality is a means employed by at least one agency in the execution of an action.

AGENCY

An agency is one of two entities:

1. An agency is a self-aware sentient being capable of action.
2. An agency is a collective of self-aware sentient beings capable of united action.

ACTION

An action is a nomologous ontogen in which no delta of any evolute constituting that ontogen is an ontogen.

In the context of this explication, an action is a nomologous ontogen satisfying both of the following two conditions:

1. No delta of any evolute constituting that ontogen is an ontogen.
2. There exists at least one fact of INTENTIONALITY which is an enablement condition of least one delta of at least one evolute constituting that ontogen.

...

INTENT DERIVATION

Intent Derivation is the strategic enterprise engineering analog of the technical systems engineering process called *requirements identification and analysis*. Like its analog, Derivation is basically an inverse root cause analysis process. Unlike its analog, Derivation is principally concerned with the analysis of *strategic* antecedence of *all* stakeholder intents, not just antecedence analysis of technical Customer intents—a very difficult and abstract undertaking indeed. The inputs to Intent Derivation are the explicated formal intents articulating stakeholder purposes, created as outputs of the Explication subprocess. The *intermediate* outputs of this process are fully instantiated and validated instances of the Antecedence and Consequence structure depicted by Figure 31 on page 100 in Attachment 2. These structures represent nominal identifications of intents whose accomplishments constitute

necessary accomplishment conditions for the input intents, called *antecedents*. The final outputs are fully instantiated and validated instances of the Architecture Element structure *for each antecedent intent*. These are generated by iterating between the Derivation and Explication processes, once for each antecedent. This iteration continues until certain specific termination conditions are reached.⁵⁶ Thus the form of this iteration suggests a ‘spiral,’ like that which is frequently used to depict the iterative execution of the analogous systems engineering processes. Here one of the crucial elements produced are definitive *MOE allocations* to the antecedent intents: these are essential for downstream action programming and enterprise assurance processes.

We were unable to complete a derivation of antecedents to any particular intent during MEREOS Phase III. However, we were engaged in developing a derivation of Focus On Total Solutions iteratively in conjunction with our explication of that intent, and again we have included an extract of from a working paper on the antecedents of TOTALITY for processes to illustrate what *this* process involves—to convey a “feel” for the formal derivation process.

Total solutions are instrumentalities for achieving all strategic purposes of all stakeholders across the entire life-cycle of a given strategic context. One special kind of total solutions are life-cycle balanced products capable of being used by a customer to satisfy all their needs in a given operational context. Producing definitions of these special kinds of total solutions is the enterprise of systems engineering. Total solutions in the LMAS enterprise operating system context are life-cycle balanced products and services for all LMAS stakeholders. Creating an enterprise operating system for producing and sustaining these is the enterprise of enterprise engineering.

Total solutions can only be realized by consummately capable agencies possessing complete and accurate information executing total processes supported by superior technologies. That is, total solutions require total processes, total technics⁴⁴, and total information. These are the three principal entailments of achieving total solutions; all others follow from these.

Focus is a conjunction of two conditions. The first is a convergence of intent among all the stakeholders in a given strategic context—that is, a sustainment of a single intent by all of them. The second is an interdiction of conflicting intents maintained by stakeholders in a given strategic context—that is, an inhibition for action based on a divergent intent held by any of them.

Focus on the central intent of a given strategic context can only be attained by consummately capable agencies possessing complete and accurate information and meta-information⁴⁵ executing total strategic processes supported by superior technology. That is, focus entails a total strategy process, total technics, and total information; all other derivative antecedent conditions follow from these three.

Total entities—total in general—are complete and optimal in the life-cycle of a given context. The condition of totality differs for processes, technics, and information.

Total Processes

There are exactly three processes constituting the life-cycle of any given entity. These are Provisioning, Execution, and Assurance. A total process satisfies at least one of the following criteria:

1. It is an optimal configuration of an entire life-cycle; that is, it comprises all provisioning, execution, and assurance for at least one entity type.
2. It is a complete and optimal configuration of at least one of the above three processes; that is, it is an optimal element of a total process.

⁴⁴ The term “technics” is used to cover both human capability *and* technology.

⁴⁵ Information about information; in this particular case, information about the information possessed by other stakeholders—especially those maintaining divergent intentions. *Competitors* are classical examples of these stakeholders, and *competitive intelligence* is a classical example of meta-information in this context.

⁵⁶ Crudely speaking, Derivation proceeds until all identified ‘leaf’ antecedents are either (i) already actual; (ii) can be actualized by an ‘atomic’ action of an enterprise operating system; (iii) cannot be actualized by any possible operating system action or process—i.e., are *scope-extrinsic*; or (iv) define the same condition as at least one other antecedent or (b) are variants of other antecedents, such that they satisfy conditions (ii) or (iii).

3. It is a complete and optimal configuration of a permissible variant of a total process or an element of a total process.
4. It is demonstrably traceable process requirement imposed either by stakeholders with governing authority over the enterprise, or by the LMAS Constitution.

If a process satisfying the fourth criterion above does not also satisfy one of the other three, then the process is *defective*—it is not total. This condition is an effectivity for a corrective action process which must as defined in the architecture and implemented by the operating system.

An enterprise operating system process is a systemic process. An enterprise realization process is a strategic process. Enterprise operating system corrective action processes are architectonic—these are special kinds of enterprise realization processes.

Total solutions require total systemic processes. Focus requires total strategic processes. Autonomism requires architectonic processes. Thus two specific antecedents of Focus On Total Solutions are:

1. Execute All Enterprise Actions Via Total Systemic Processes
2. Govern All Enterprise Actions Via A Total Strategy Process.

Both of the above entailments can be stated by the following general imperative:

Act Via Total Processes.

...

INTENT SYSTEMIZATION

Intent Systemization is the strategic enterprise engineering analog of the technical systems engineering process called *system optimization*, applied to stakeholder intents rather than definitions of solutions to those. These two processes are however only functional analogs—they have almost nothing in common other than their roles in their respective parent processes.

We have stated before that purposes—and thus the intents that articulate them—are numerous and diverse in kind. One side effect of the Intent Derivation process is to increase both the quantity of and the diversity among these intents, as antecedents are iteratively identified and explicated. In an ideal universe where capabilities and resources were infinite, enterprise objectives would be identical to stakeholder intents, and achieving these objectives would represent accomplishment of all the intents of all stakeholders. But of course this ideal is an absolute empirical impossibility and almost certainly a formal one, given typical divergences among stakeholder intents—those which are diametrically opposed being paradigm examples. Enterprise objectives will *never* be identical to intents. For exactly these same reasons, neither will enterprise moes ever be identical to stakeholder moes. As a practical matter of fact, gaps will *always* exist between what is desired and what is achievable.

In the final analysis, there are only four fundamental ways to bridge such gaps. The first is by invention—enhancing the effectiveness of existing capabilities or creating new ones. The second is by acquisition—enhancing utilization efficiencies of existing resources or securing additional resources, via procurement, exchange, or alliance. The third is by exclusion—eliminating one or more stakeholder intents from the strategic mix. The fourth is by *systemization*, which involves *optimizing the objectives themselves in advance* of defining action programs for carrying them out. That is, cogent intent systemization can, in many circumstances, *pre-empt* the need to devise actions to overcome gaps entailed by intents that can be factored or subsumed, and can significantly decrease the amplitude of actions required to overcome those that remain. This last technique is a signature of expertise in the art of politics and a hallmark of incisive strategic skill. The Intent Systemization process is a formalization of this technique.

The primary inputs to the Intent Systemization process are the validated outputs of the Intent Explication and Derivation processes—that is, precisely stated intents, together with their anteceded-

ents, and MOES. The *intermediate* outputs of this process are formal analyses of taxonomic similarities and divergences among the these, developed using a self-applicative taxonomy of SIMILARITY itself.⁵⁷ Fulfilling the analogous function of product/process structure optimization in a strategic management context, these analyses are accordingly represented using exactly the same relations as those used in that context—namely, ARTICULATION, FACTORIZATION, and UNIFICATION⁵⁸ relations. Here they relate elements of *intentional* structures, respectively delineating Realization process segmentations, uniformities, and conjunctions among these. The final outputs of this process are balanced, actionable, and maximally dense enterprise objectives and MOES, defining the requisite outcomes of enterprise actions.⁵⁹ These objectives subsequently undergo achievability analysis via the Operational Analysis process, one of three major subprocesses constituting the Action Programming process. A primary formal structure involved in that process, called a Strategic Context, is briefly described in paragraph 4.2.4.4 below. Instances of these structures in turn are the primary inputs to other downstream Action Programming subprocesses, and to the Solution Realization and Enterprise Realization processes.

4.2.4.3 Effectiveness Measure Systematics

Qualification criteria must be explicitly defined to enable governance of enterprise actions, and to enable determination of the value of solutions for instrumentalizing the actions of enterprise stakeholders. These criteria *are* MOES—and defining, imposing, and determining their states constitute a core enterprise process called *Assurance*. As we stated earlier, MOES are internalized stakeholder value impositions on the enterprise—they define *what* value to *which* stakeholders are to be delivered via execution of enterprise processes. Thus defining an enterprise operating system and an assurance process for that system entails a systematic and definitive answer the following question:

What are appropriate performance measures for each of the enterprise processes?

Our efforts to answer this question were incomplete at the end of MEREOS Phase III, although a great many results had been obtained and interim documentation produced by the end of calendar 2000. The overall objective of these efforts is to specify a suitable *mensuration system* for the NAE operating system—a critical mechanism of its Assurance process. We have decided not to incorporate the text of these working papers into this document, as they are exceedingly abstract, employ the technical language of our metasytematics, and thus would be of little value to convey concrete results. We have instead summarized the specific areas we have focused on below.

■ Measure Of Effectiveness Delimitation

Determining specific and appropriate performance measures for enterprise processes presupposes a prior and formally adequate generalized definition (taxonomic delimitation) for the class MEASURE OF EFFECTIVENESS itself. No such delimitation exists, so we were forced to undertake its development.

■ Qualification and Quantification Process Definitions

Imposing MOES on a process presupposes that processes exist for determining the values of the magnitudes in question. Accordingly we developed formal definitions of QUALIFICATION and QUANTIFICATION—the processes required to determine the values of qualitative and quantitative MOES, respectively. The structures and their elements associated with these processes—designated METROLOGY and NOMOLOGY—are depicted in the Architecture Element Elements drawing on page 99 in Attachment 2.

⁵⁷ The SIMILARITY taxonomy is depicted by Figure 32 on page 101 in Attachment 2.

⁵⁸ Definitions of the ARTICULATION, FACTORIZATION, and UNIFICATION relation types are presented in paragraphs 3.2.4.1, 3.2.4.2, and 3.2.4.3, respectively, in Section 3. Here the entities standing in the attribute positions of these relations are intents and their elements, rather than product structures and their elements.

⁵⁹ In many cases, these objectives can simply be ‘read off’ the inter-intentional relations. That is, once the latent discontinuities (ARTICULATION relations), intrinsic homologies (FACTORIZATION relations), and attainable subsumptions (UNIFICATION relations) among a given set of intents and their relata have been delineated, restating these as enterprise objectives coordinate to these relations is a reasonably trivial exercise.

I MOE Taxonomy

There are dimensions of variation among enterprise processes that partially determine the appropriateness of specific MOES for those processes. For example, the Strategy Realization process and most of its secondary subprocesses are *abstract* and synthetic, while Solution Realization and most of its secondary subprocesses are *concrete* and synthetic. MOES which are appropriate for abstract cognitive processes are not appropriate measures for concrete physical ones. There are several other variations among processes over and above these two, all of which have implications for MOE imposition. There are also *intrinsic* variations among MOES themselves. For example, systems engineers correctly distinguish between *performance* and MOES expressing requisite degrees of this attribute⁶⁰ and *effectiveness*, and MOES expressing requisite degrees of that one. Without getting bogged down in a long discussion, suffice it to say for our purposes here that Performance is *efficiency* of Realization and Effectiveness is *capability* of Realization—these are very distinct characteristics, indeed.

One of the more difficult problems involved in developing a formal taxonomy of MOES for enterprise processes is identification of a *suitable* set of *basic magnitudes*. If we were embarked on a project to develop such a taxonomy in a more concrete domain, this would not be a problem. For example, there are several families of basic physical magnitudes to choose from—such as those employed in the ‘metric’ (SI) system—and there are already a large number of ‘intentional’ magnitudes that must be accommodated in any undertaking of this kind—the gamut of economic measures being paradigm examples. But while magnitudes such as duration and economic efficiency may be important enterprise process MOES, at least to *some* stakeholders, they and their kin are far from the *only* ones of significance from an assurance standpoint. Thus the question is: if physical and economic magnitudes constitute only a subset of the magnitudes involved to determine the performance and effectiveness of enterprise processes, what *are* the others? And perhaps even more importantly, given any particular answer to that question, how do we know we have identified all the *possible* and *relevant* magnitudes? That is, how can we be sure that the list is complete, systematically derived, and apposite to the purposes at hand?

I Purpose/Process Taxonomy

What constitutes an acceptable degree of effectiveness or performance is always relative to a specific purpose—i.e., a particular stakeholder type. For example, MOES appropriate for determining requisite degrees of Customer value are simply not the same as those for determining requisite degrees of Shareholder value. This entails that the answer to the taxonomic question at hand will be a ‘vector’ of MOES for each process, each element thereof representing the appropriate measure for a particular stakeholder type with respect to that particular process. There are 10 principal stakeholder types, and 21 enterprise processes. This entails that there are in principal *at least* 210 MOES for the enterprise processes. Developing this taxonomy is a decidedly non-trivial endeavor.

4.2.4.4 Operational Analysis and Strategic Contexts

A key requirement identified in paragraph 4.2.3 above is that mission definitions suitable for enterprise operating system synthesis must consist in formal and effective expressions of strategic intent. Defining invariant purposes in actionable form, these constitute the origins of all ‘top-level’ enterprise operating system requirements. Indeed, systematic operating system design, deployment, and operation mandates that *all* stakeholder purposes must be formalized, invariant or otherwise. Thus Intent Formalization is a critical enterprise operating system process.

Accomplishing stakeholder intents requires effective action. Effective action by enterprises via their operating systems presupposes prior definition, and these definitions—action *programs*—are the primary outputs of the Action Programming process. Action Programming—i.e., enterprise operating system programming—is the Development phase of the Strategy Realization process, depicted by the middle cell in the top row of the diagram in Figure 19. The principal inputs of this process are enterprise objectives and MOES defined by the Intent Systemization subprocess of Intent Formalization. Stipulating requisite outcomes of enterprise actions, these objectives and MOES con-

⁶⁰ These are called *Technical Performance Measures* (TPMs) in systems engineering.

stitute the governing functional, performance, and effectiveness requirements for synthesis of action programs to achieve them.

Like its analog in product definition contexts, action program synthesis must also be governed by additional constraints over and above the functional, performance, and effectiveness requirements represented by enterprise objectives and MOEs. One of the most crucial of these is *executability*—the analog of *producibility* in product definition contexts. That is, an action program that cannot be executed is just as worthless in a Strategy Realization process context as an unproducible design is in a Product Realization process context. And, just as producibility determinations entail prior evaluations of extant and envisioned technical capabilities, ascertaining the executability of action programs necessitates prior evaluations of existing and postulated strategic capabilities, as those bear on accomplishing a given objective. Thus definitive determinations of *feasibility* or *achievability*—the same thing expressed as an attribute of objectives and intents—constitute crucial enablements of action program synthesis and subsequent executability evaluations of those programs.

Achievability determinations are products of the *Operational Analysis* subprocess of the Action Programming process. These determinations are rendered as fully instantiated instances of the Strategic Context structure, graphically depicted by Figure 22 below. This structure is a variant of our

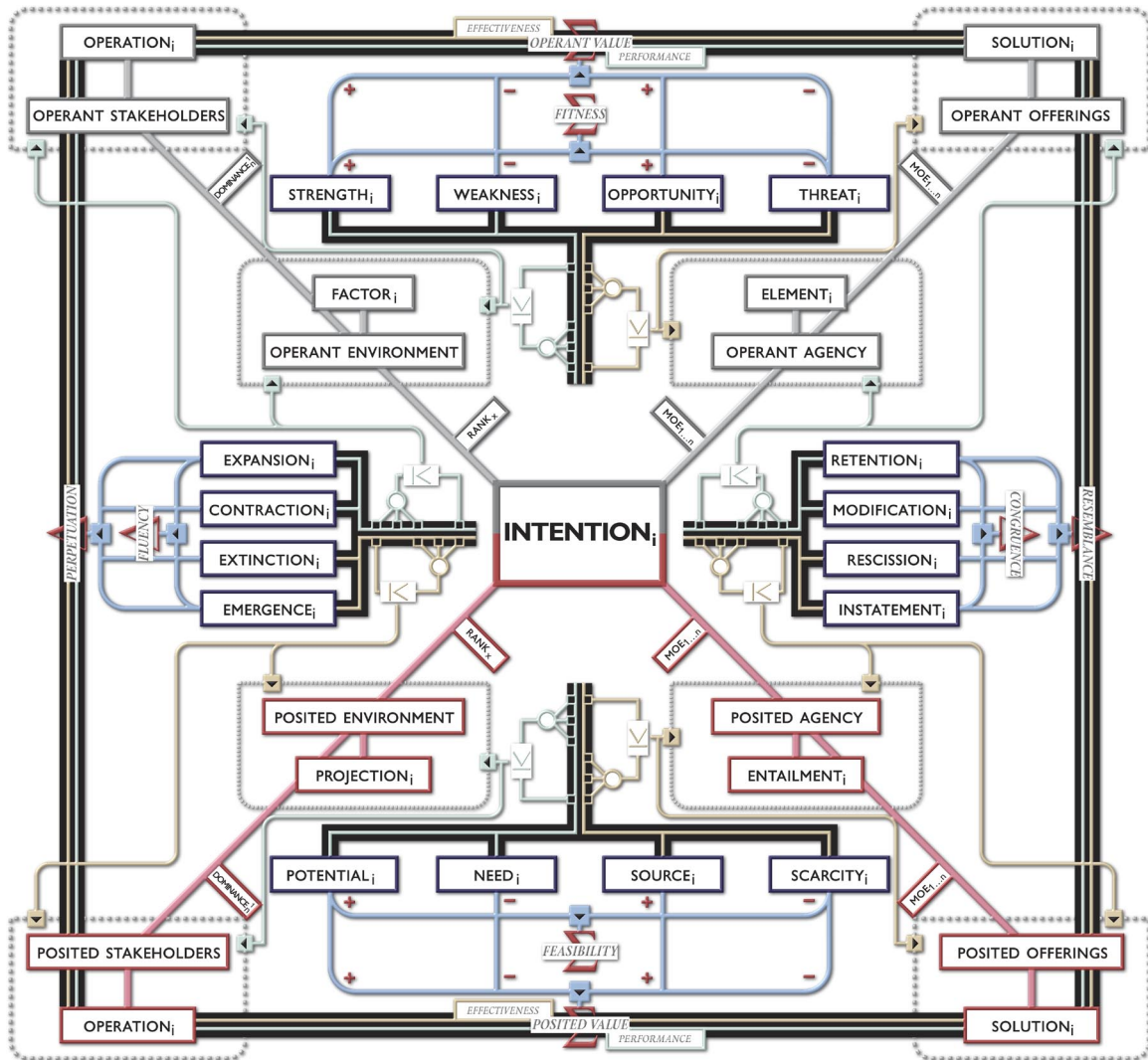


Figure 22. Strategic Context Elements

INTENTIONAL COMPLEX metastructure, adapted to support the formal needs of the Operational Analy-

sis process and other downstream Action Programming subprocesses—specifically, the Program Synthesis subprocess. The two primary functions of the Strategic Context structure are to inform achievability analysis of stakeholder intents and their correlate enterprise objectives, and to frame the results of that analysis. This structure is, accordingly, a formalized and extended realization of the implicit structures employed in the swot⁶¹ and capability assessment processes in strategic management.

■ Classification Scheme

The process of developing rigorous and detailed characterizations of strategic intent via the Intent Formalization process is inherently difficult. Precise analyses framed in terms of complex abstractions are required to achieve meaningful results. Carrying out operational analysis to determine the practicability of accomplishing these intents is even more difficult. Intrinsically complex material matters of fact, such as enterprise capabilities, environmental factors, and certain specific relationships between these—both actual and implied—are principal subjects of such analysis. Nevertheless, insuring that results of enterprise actions are congruent to stakeholder purposes is the paramount aim, regardless of the difficulties involved, and so operational analysis must also accordingly insure that congruence of actions to intents is consistently maintained.

Another factor exacerbating the difficulty of operational analysis is the sheer constitutional complexity of the entities themselves. Agencies—enterprises and their operating systems—environments, stakeholders and their actions, and the solutions necessary to instrumentalize those, are very complex *systems*; that is, each one individually is an integral product-process gestalt and is typically both taxonomically diverse and quantitatively extensive. And, the relationships among these are correspondingly diverse and numerous as well. However, only a few element types, relations, and attributes of these entities are relevant to this process; the full systemic power of the Architecture Element structure taxonomy is simply not required for purposes of operational analysis. This fact can be exploited to mediate the difficulties involved by *redacting* the Architecture Element structure into a more succinct structure for analytic purposes. The Strategic Context structure is the result.

As the primary aim of operational analysis is to ascertain the achievability of a given intent, the central object in a Strategic Context structure instance is the intent itself, as depicted by the box labeled INTENTION_i in the center of Figure 22. All strategic intents objectify or designate a specific state or states of affairs (soas) which an agency or agencies desire to be actual. From a purely formal point of view, every intentional complex comprises the following elements:

1. the intention itself;
2. the soa or soas that intention objectifies;
3. the agency or agencies sustaining the intention; and,
4. the environmental context in which the sustainment of that intention occurs.

The Strategic Context structure *eliminates* the distinction between the intent and the soa it designates—they are conceived as one object for operational analysis purposes. Eliminating this distinction is one step in redacting the Architecture Element structure into the Strategic Context structure. This is why there is only one element of that type in the structure—i.e., INTENTION_i.

EFFECTIVITY

The formal distinction of *effectivity* between the actual intent and the desired state it objectifies is *not* eliminated. Effectivity is instead employed as a ‘dimension’ of the Strategic Context structure. The two ‘coordinates’ of this ‘axis’—*operant* and *posited*—are used in conjunction with others to ‘grid’ Strategic Context elements and, thereby, to delineate distinctions between them which are significant for operational analysis. The Strategic Context diagram depicts these differences in effectivity both visually and by some element designations. Specifically, the top half of the box labeled

⁶¹ Strength, Weakness, Opportunity, and Threat. An excellent discussion of these four elements in a pure strategic management context can be found in David’s *Strategic Management Concepts*; see our references 5.2 and 5.3.

INTENTION_i is grey, while the bottom half is red, implicitly depicting the location of the axis itself.

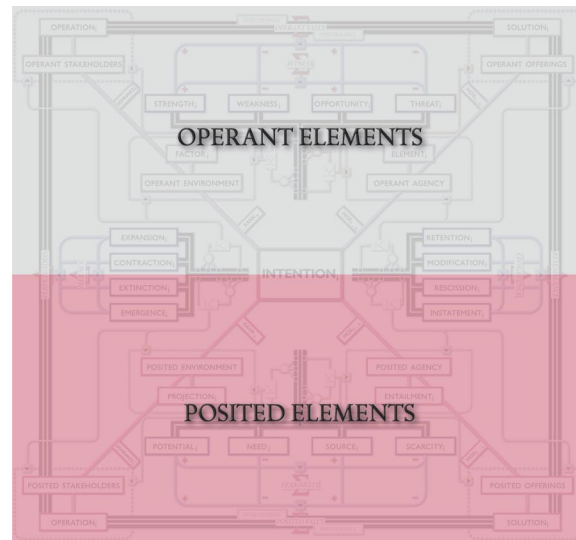


Figure 23. Effectivity Coordinates

Hence the top half of the drawing contains those elements which are **OPERANT**—i.e., actual elements of the agency sustaining the central intent and those of its situating environment. The bottom half contains those which are **POSITED**—i.e., postulated elements of the sustaining agency and its environment projected by that intent. We will describe these in more detail shortly.

LOCATION

As we stated before, all intents are situated in specific environments, and this locative feature is employed as the second 'dimension' of the Strategic Context structure. The two coordinates of this axis—*adjunct* and *advenient*—are used to distinguish Strategic Context elements which are extensions of intents from extrinsic elements comprising them. One can envision this axis an imaginary vertical line through the center of the Strategic Context diagram, with adjunct elements positioned on the right and environmental elements on the left.

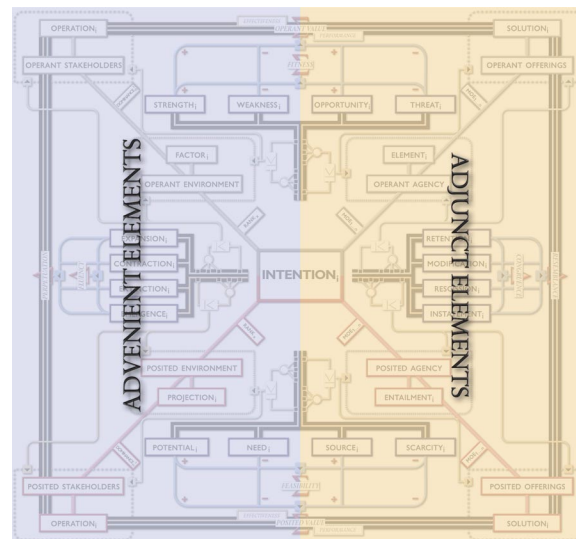


Figure 24. Location Coordinates

Combining both the **EFFECTIVITY** and **LOCATION** dimensions, we obtain four 'quadrants,' each con-

taining a specific category of Strategic Context element types.

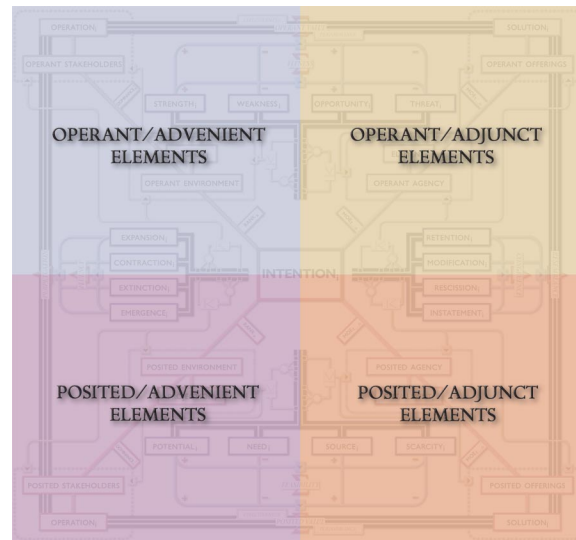


Figure 25. Element Type Quadrants

I Elements

INSTRUMENTALIZATION AND APPLICATION RELATIONS

Achieving an intent is an outcome of *effective actions* or *processes*. That is, realizing a posited SOA projected by a desiderative intent entails execution of an action or a process that results in at least one of the following:

1. *that* SOA becoming actual;
2. an *instance* of that SOA scheme becoming actual; or,
3. an *admissibly congruent* SOA becoming actual.

Any intents necessitating the effort of operational analysis cannot be directly achieved by atomic actions. They entail complex realization processes to achieve them and correspondingly complex instrumentalities—i.e., *solutions*—to provision the execution and governance of these processes. Realization of the instrumental solutions in turn entails agencies with the capabilities required to render them, and typically these are different agencies than those executing intent realization processes. That is, one agency typically executes one or more operations to realize the intent, thereby performing the role of *effective agent* in a given strategic context. Another agency executes the entailed solution realization process, thereby performing the role of *instrumental agent* in that context. Even if the same agency fulfils both the effective and the instrumental roles in a given strategic context, that agency will exemplify distinct characteristics with respect to each of those roles. Moreover, its relations to the remaining elements constituting that context will accordingly differ, and so will the attributes of interest for operational analysis it exemplifies in each of those roles. Again however, these roles are typically performed by distinct agents.

In light of the above remarks we can now introduce the two groups of Strategic Context elements—the relation types INSTRUMENTALIZATION and APPLICATION, their attributes, and the entity types standing in those attribute positions. These two relations are respectively the functional and dispositional variants of the CAPACITATION relation type, defined in paragraph 3.2.5.2 in Section 3. INSTRUMENTALIZATION, the relation of agential provisionment, is the formal structure invoked in our characterization of the ENTERPRISE taxon in paragraph 4.2.4.1, and is consequently a primary relation among the three ‘top-level’ enterprise processes, as illustrated by the diagram in Figure 19. OPERANT and POSITED variants of this relation, together with the entity types in those attribute positions, comprise a major group of Strategic Context elements as well. The diagram in Figure 26 below

illustrates this intersection between the Strategic Context structure and the Enterprise Process Taxonomy by overlaying the elements they share.

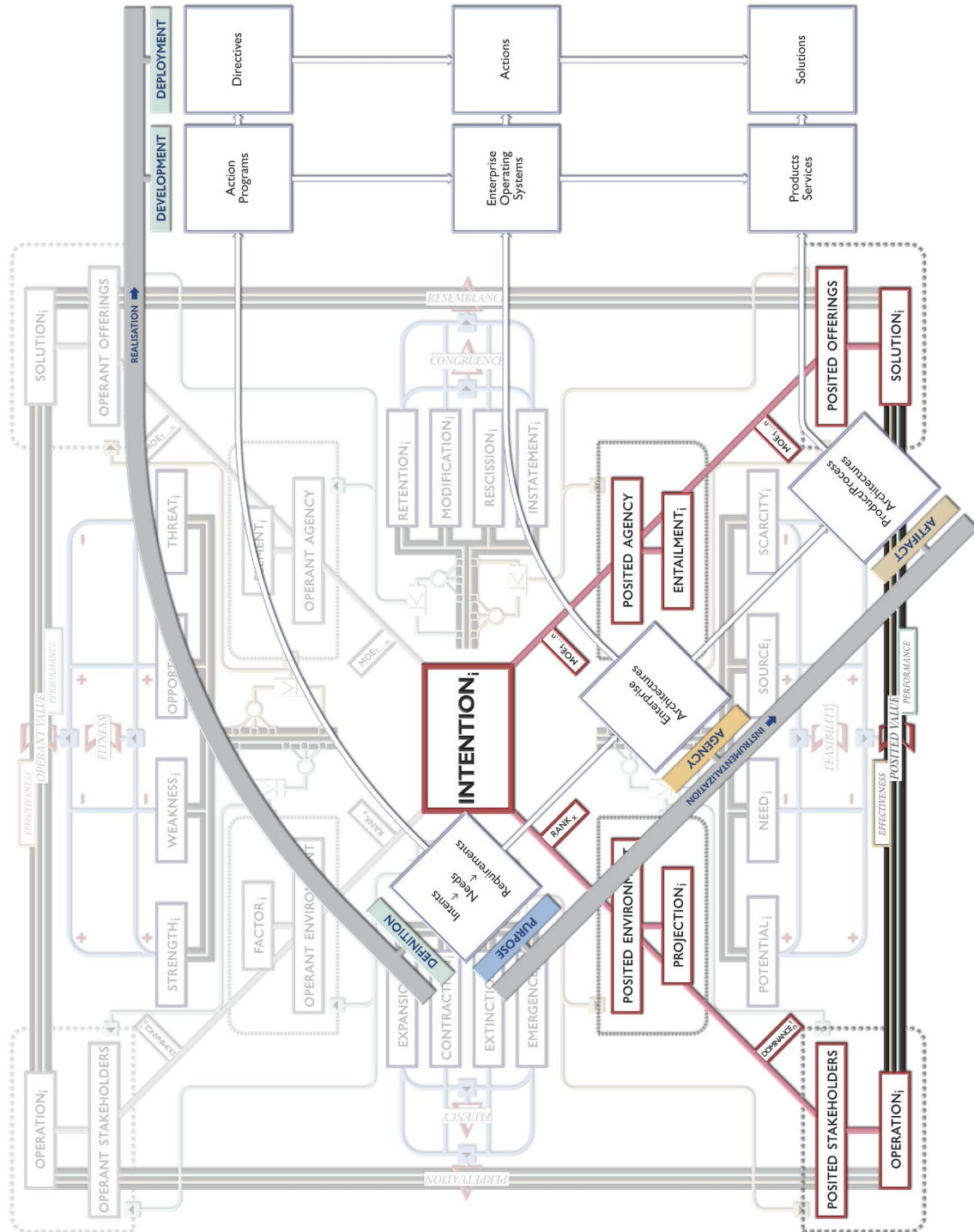


Figure 26. Strategic Contexts and Enterprise Process Definition Phases

Note that only the *posited* INSTRUMENTALIZATION relation and attribute elements are shown unghosted in this diagram. The red 45° line drawn through the upper right quadrant in the above figure, connecting the boxes labelled INTENTION_i, POSITED AGENCY, and POSITED OFFERINGS, depicts the POSITED variant of this relation and its elements. Its OPERANT counterpart and elements are depicted along the

grey 45° line drawn through the upper left quadrant above (upper right in the normal orientation); these can be seen more easily in Figure 22 than in the above diagram.

APPLICATION, the relation intrinsic provisionment, is the relation bearing the determining attributes of stakeholder value, as was illustrated by the diagram in Figure 20. Both OPERANT and POSITED variants of this relation, together with the entity types in its attribute positions, comprise a second major group of Strategic Context elements. The black line labelled OPERANT VALUE in the above figure, which connects the boxes labelled OPERATION_i and SOLUTION_i, depicts the POSITED variant of this relation and its elements. Again, its OPERANT counterpart and the elements of that APPLICATION variant, which appear opposite to its POSITED counterpart, can be seen more easily in Figure 22 than in the above diagram.

ENVIRONMENTS

Again, all intents are situated in specific environments, and so are both the effective and instrumental agencies that sustain them. Environments, their elements, stakeholder agencies, and the operations they execute to achieve their aims accordingly constitute another group of Strategic Context element types.

STRATEGIC MAGNITUDES

The remaining Strategic Context structure elements are *magnitudes* collectively determining intent achievability. These magnitudes fall into two major categories—*Contrast* and *Capacity for Effective Action* (CEA).

As we pointed out earlier, gaps will inevitably exist between what stakeholders desire and what enterprises can actually achieve. While many of these can be abated in advance via the Intent Systemization process, some will almost certainly remain. On the other hand, capability and resource surfeits will frequently exist as well. Identifying these gaps and surfeits and characterizing them in terms of these two categories of magnitudes is the main focus of operational analysis.

CONTRAST

The operant and posited configurations of Strategic Context elements relative to a particular intent typically differ. Thus the achievability of an intent by the operant configuration of an enterprise in its operant environment will almost certainly diverge from the achievability of that intent by the entailed or posited enterprise configuration in the projected environment. Any such divergences will have ramifications for both capabilities and resources. Accordingly, one task required to establish intent achievability involves determining two correlated sets of values expressing these differences. The first set comprises Contrast values between operant and posited configurations of entities in INSTRUMENTALIZATION attribute positions—namely, enterprise and solution configurations. These are called *Congruence* and *Resemblance* values, respectively, and are depicted above the “delta” symbols on the right side of Figure 27 below. The second set consists in Contrast values between operant and posited environment configurations situating the intent and stakeholder operations. These are called *Fluency* and *Perpetuation* values, respectively, are depicted above the “delta” symbols on the left side of Figure 27.

CEA

CEA also consists in two distinct but correlated sets of values. The first set, called *instrumental* CEA, comprises CEA values representing the abilities of instrumental agencies to render solutions required to provision accomplishment of the intent. The second set, called *functional* CEA, comprises CEA values representing the abilities of stakeholders sustaining the intent to achieve it and thus other superordinate intents, via operations employing enterprise solutions. Both of these sets are summations of operant and posited values of the same attributes. The operant and posited values of instrumental CEA are called *Fitness* and *Feasibility*, respectively, depicted above the “sigma” symbols on the top and bottom of Figure 28 below. The operant and posited values of functional CEA are called both called *Value*—i.e., the attribute of the Capability relation defined in paragraph 4.2.4.1. Determining these two sets of CEA values is the primary product of Operational Analysis.

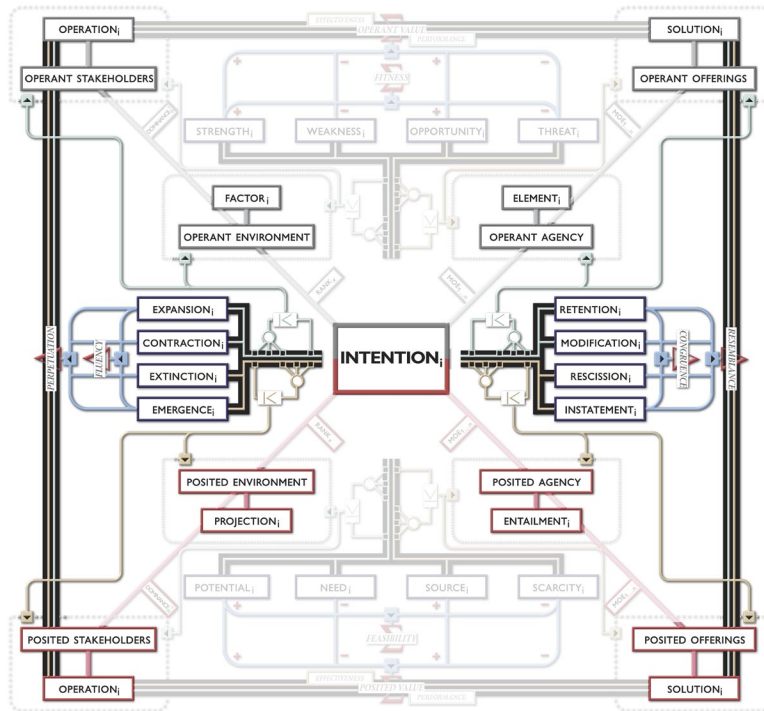


Figure 27. Contrast Variants

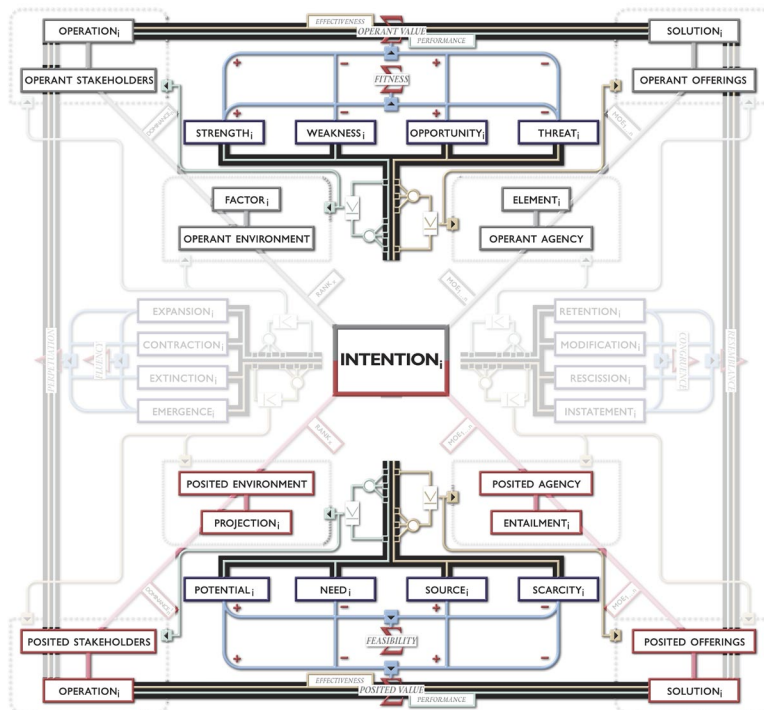


Figure 28. CEA Variants

“Every tool carries with it the spirit by which it has been created.”

Werner Karl Heisenberg, 1958

5

CONCLUSIONS

Product data are the premier informatic capital of industrial enterprises, and the overall quality of these assets—their completeness, accuracy, accessibility, and maintainability—are major determinants of enterprise performance and vitality. The processes and supporting infrastructure that produce and sustain these data are thus critical and systemic elements of enterprise operating systems, and endeavors aiming to improve the effectiveness of either of these are directly focused on enhancing the value enterprises can deliver to their stakeholders.

There is certainly no dearth of technical, strategic, or socio-economic obstacles impeding realization of substantive gains in product representation process effectiveness and infrastructure capability. As one informed Air Force lab director once dryly observed, “It’s a target-rich environment.” In our view two of these impediments stand out with a certain extreme acuity. The first is very technical indeed— it is the metasystematic deficiency that lies at the heart of both the representational shortcomings of existing information systems we discussed at length in Section 3, and the methodological gap we described in Section 4. The second obstacle is essentially socio-economic in nature, although there are strategic overtones to it as well. This impediment consists in the widespread diminishment in understanding of product representation process and infrastructure requirements in industrial organizations, their consequent abrogation of control over those to commercial software vendors, and the resulting albeit unsurprising conflation of total process solutions with information systems.

Existing product representation processes in industrial enterprises are segmented to align with the Definition, Development and Support subprocesses of Product Realization processes. This partitioning compartmentalizes the organizational and information systems infrastructure elements of these processes as well—the famous ‘wall’ between engineering and manufacturing organizations and the well-known difficulties encountered by those attempting to integrate PDM and ERP systems being two particularly illustrative cases in point. These divisionalizations are have a long historical

standing in industry and are, therefore, familiar to all and accepted as the natural order of things by many. Nevertheless, they are artificial, suboptimal, and in light of both the product representation and enterprise operating system needs previously described, they are outdated and constitute a serious impediment to meeting those needs.

Recognition by many industrial enterprises of problems attributable to unnecessary and excessive divisionalization was a major impetus for their enthusiastic adoption of the Integrated Product Development (IPD) model of the Product Realization process and Integrated Product Team (IPT) organizations. And, those enterprises that implemented this inherently integrative approach achieved significant gains in Product Realization performance and effectiveness as a result. However, achieving the full potential of large-scale enterprise process integration entails suitably capable infrastructure over and above the organizational element. Specifically, realizing that potential and sustaining the resulting gains entails equally integrative information systems—and such systems just do not exist today. Moreover, such sweeping changes in an enterprise's Product Realization process entails commensurate and carefully coordinated architectural changes to all its other major enterprise processes—and the requisite methodological technics to accomplish that with a reasonable certainty of success do not exist either. Our objective was to bring these tools closer to reality than they were at the beginning of the MEREOS program.

The adolescence of enterprise engineering as a discipline constitutes a significant strategic opportunity for an innovative academic organization to establish itself as a pre-eminent provider of research, practical applications, and graduates in this interdisciplinary field. In pursuit of that opportunity, an academic institution would need to conduct focused research to fill gaps in enterprise engineering principles and methods, thereby strengthening the foundations of that discipline, enabling the eventual codification of its methods, and provisioning actions to fulfill the needs we have presented here.

The Air Force may at some point find it in their interests to play a role in such an endeavor as well. One potential role would comprise a second iteration of its most successful industrial base program—the Integrated Computer-Aided Manufacturing (ICAM) program—focused this time on fostering development of solutions to the needs currently confronting the aerospace and defense industry, some which we summarized in this document.⁶²

⁶² The ICAM program was veritable fountain of novel and valuable technology. For example, it created the first full-scale idealized enterprise architecture, called the Factory of the Future (FoF). ICAM also funded and managed development of a plethora of technologies designed to satisfy FoF architecture requirements. Notable among these were the ICAM Definition Language (IDEL), the Integrated Information Support System (I²S), the second attempt to build a 3-schema DBMS), the designs for the IPD process mentioned above, the Product Definition Exchange Standard (PDES, which later became STEP), advanced Machining, Composites, and Assembly centers, and, late in the program, the first experimental version of our own PACIS system.

REFERENCES

- 1] American National Standards Institute (ANSI).
Committee on Computers and Information Processing (X3)
Standards Planning and Requirements Committee (SPARC), Data Base Study Group (DBSG).
DBMS FRAMEWORK REPORT, ANSI/X3/DBSG REPORT 1975.
- 2] James F. Cox and John H. Blackstone Jr., Editors.
APICS DICTIONARY.
American Production and Inventory Control Society (APICS), Falls Church, VA, 1998.
ISBN 1-55822-123-9.
- 3] Benjamin Blanchard and Wolter Fabrycky.
SYSTEMS ENGINEERING AND ANALYSIS.
Prentice Hall, 3rd. Edition.
ISBN 0-13-135047-1.
 - 3.1] Section 1.5 Technology and Technical Systems, page 13.
 - 3.2] Section 2.6 Implementing Systems Engineering, pages 36-37.
 - 3.3] Section 2.2 System Life-Cycle Engineering, page 19.

The definitive textbook on systems engineering, period.

- 4] Derek Channon, Editor.
ENCYCLOPEDIA OF STRATEGIC MANAGEMENT.
Blackwell Publishers, Inc., Oxford, UK, 1997.
ISBN 1-55786-966-9.
 - 4.1] pages 171-172.
- 5] Fred R. David.
CONCEPTS OF STRATEGIC MANAGEMENT.
Prentice-Hall, Inc. 1999, Seventh Edition.
ISBN 0-13-080784-2.
 - 5.1] "Integrating Intuition and Analysis," Page 6.
 - 5.2] "The External Factor Evaluation (EFE) Matrix," Page 129.
 - 5.3] "The Internal Factor Evaluation (IFE) Matrix," Page 165.

An concise but very comprehensive textbook on the topic, and an outstanding introduction to the field.

- 6] David Fitzpatrick.
Quotation in *Effective Application of 'Lean' Remains Disappointing*.
Aviation Week & Space Technology.
January 22, 2001, Page 60.
- 7] Risieri Frondizi.
WHAT IS VALUE?
Open Court Publishing Co., 1971, 2nd edition.
LCN 70-128196.
- 8] GUIDE International Corporation.
REQUIREMENTS FOR A DATA BASE MANAGEMENT SYSTEM.
GUIDE Secretary Distribution (GSD) 23, 1971.

This document is the original specification for the 3-schema database systems architecture, and thus it is the progenitor of all the subsequent texts on that topic, such as the "Orange Book" produced by ANSI in 1975 (reference [1] above) and many others, such as those cited in references [12] and [15] below.

- 9] Michael Hammer and James Champy.
REENGINEERING THE CORPORATION: A MANIFESTO FOR BUSINESS REVOLUTION.
HarperCollins Publishers, Inc. 1993.
ISBN 0-88730-640-3.
9.1] page 35 and 39.
- 10] Ronald Henkoff.
BOEING'S BIG PROBLEM.
Fortune Archives.
(ISSN 0015-8259) Vol. 137 No. 1 page 964.
- 11] Institute of Electrical and Electronics Engineers, Inc. (IEEE).
IEEE STANDARD FOR APPLICATION AND MANAGEMENT OF THE SYSTEMS ENGINEERING PROCESS
P1220-1998.
- 12] International Organization for Standardization (ISO).
CONCEPTS AND TERMINOLOGY FOR THE CONCEPTUAL SCHEMA AND THE INFORMATION BASE.
ISO/TR 9007(E), 1987.
- 13] International Organization for Standardization (ISO).
STANDARD FOR THE EXCHANGE OF PRODUCT MODEL DATA (STEP).
ISO 10303.
- 14] Frank Ireson, Claude F. Coombs, Jr., Richard Y. Moss.
HANDBOOK OF RELIABILITY ENGINEERING AND MANAGEMENT.
McGraw-Hill, 1995, 2nd Edition.
ISBN 0-07-12750-6.
- 15] D. A. Jardine, Editor
THE ANSI/SPARCDBMS MODEL.
North-Holland Publishing Company. 1977
ISBN 0-7204-0719-2.
- 16] Tadeusz Kotarbinski.
KOTARBINSKI'S PRAXIOLOGY.
Edited by Peter Dudley.
Centre for Systems Studies Press, Hull, UK, 1995.
ISBN 0-85958-847-5.
Originally published in Polish in 1955 as *Traktat o dobrej robocie*.
First published in Great Britain by Pergamon Press, 1965 as:
PRAXIOLOGY: AN INTRODUCTION TO THE SCIENCES OF EFFICIENT ACTION.
Translated by Orlin Wojtasiewicz.
- 17] Roger Lincoln, Geoff Boxshall, Paul Clark.
A DICTIONARY OF ECOLOGY, EVOLUTION AND SYSTEMATICS.
Cambridge University Press, 2nd Edition, 1998.
ISBN 0-521-59139-2.
An essential resource for anyone interested in biological systematics.
- 18] Ernst Mayr and Peter D. Ashlock.
PRINCIPLES OF SYSTEMATIC ZOOLOGY.
McGraw-Hill, 2nd Edition, 1991.
ISBN 0-07-041144-1.
18.1] page 2.
A positively invaluable introduction to biological systematics. Now out of print and hard to find, but well worth it.

- 19] John R. Searle.
THE REDISCOVERY OF THE MIND.
MIT Press, 1992.
ISBN 0-262-19321-3.
19.1] Chapter 8, page 193, 2nd paragraph.
- 20] Peter M. Simons.
PARTS.
Clarendon Press, Oxford, 1987.
ISBN 0-19-824954-3.
20.1] Page 27.
20.2] Page 28.

The definitive text on part-whole relations.
- 21] Peter Simons and Charles Dement.
ASPECTS OF THE MEREOLGY OF ARTIFACTS.
Essays in Formal Ontology.
Poli, R. and Simons, P. (Editors).
Kluwer Academic Press 1995.
- 22] David Woodruff Smith.
THE CIRCLE OF ACQUAINTANCE.
Kluwer Academic Publishers, 1989.
ISBN 0-7923-0252-4.
22.1] Chapter II, pages 70-107.

The definitive text on the fundamental structures of intentionality and representation.
- 23] Norman Walsh and Leonard Muellner.
DOCBOOK - THE DEFINITIVE GUIDE.
O'Reilly & Associates, 1999.
ISBN 1-56592-580-7.
- 24] Alfred North Whitehead.
PROCESS AND REALITY.
Corrected Edition, edited by David Ray Griffin and Donald W. Sherburne.
The Free Press, New York, 1978.
First published by Macmillan, 1929.
ISBN 0-02-934570-7.

The magnum opus of one of the two or three greatest philosophers of the 19th and 20th centuries, and the source from which the foundations of our own formal system is derived. Exceedingly difficult—in fact almost impenetrable even for professional philosophers—the book represents a monumental and unique achievement in the history of thought.
- 25] E.O. Wiley, D. Siegel-Causey, D.R. Brooks, V.A. Funk.
THE COMPLEAT CLADIST.
University of Kansas at Lawrence, Museum of Natural History, 1991.
Special Publications No. 19.
ISBN 0-89338-035-0.

Slanted heavily towards the cladistic conceptions of taxonomy and classification, this text is nevertheless a valuable aid to understanding the fundamental concepts of biological systematics. A downloadable version in PDF format of this document is available at <http://www.nhm.ku.edu/cc.html>.



ATTACHMENTS



ATTACHMENT 1: COMMENTS CONCERNING STEP

This attachment presents an overview and some comments on STEP, developed mid-way through MEREOS program execution. We conducted this review in order to determine the representational depth of those of that standard's meta-level structures pertaining to our interests in the MEREOS program context. Current information about STEP can be found on many websites. <<http://www.nist.gov/sc4/>> and <<http://pdesinc.atcorp.org/>> are reasonable starting points.

The S**T**andard for the Exchange of Product model data (STEP) is an officially-sanctioned standards development project conducted by iso tc184/sc4/wg3, the International Organization for Standardization, Technical Committee 184 (Industrial Automation Systems)/Subcommittee 4 (Industrial Data and Global Manufacturing Languages)/Working Group 3 (Product Modeling). The actual STEP standard is iso 10303 Industrial Automation Systems - Product Data Representation and Exchange. The standard is comprised of many parts, each addressing a particular aspect of product modeling. Different parts of the standard are currently at various levels of review and approval as international standards.

STEP is described as a conceptual specification that forms a basis for communicating product information at all stages in a product life cycle, covering all aspects of product description and manufacturing specifications. STEP is intended to support approaches such as concurrent engineering and integrated product/process development. The STEP standard addresses the data required to develop, analyze, manufacture, document and support products ranging from mechanical products to electronic products, and from ships and airplanes to factories and office buildings. There are four primary components to STEP:

1. The EXPRESS data modeling language used to define all STEP data;
2. The actual schemas or definitions (i.e., data models) of STEP data, including product geometry, topology, shape, representation, features, tolerancing, structure and process specifications;
3. The application programming interface, called SDAI (STEP Data Access Interface), which is a standard interface to enable applications to access and manipulate STEP data; and
4. The STEP Physical File, which is (initially) an ASCII format file used for data exchange between or among systems.

The STEP standard is envisioned as being implemented at four levels over time. The levels are: Level 1 ASCII Text File Exchange; Level 2 Working Form Exchange or Application Program Interface; Level 3 Shared Database; and Level 4 Knowledgebase. Most of the current work is directed at levels 2 and 3.

The primary part of iso 10303 — the STEP standard — related to the topic of product structures (including BOMs) is iso 10303-44 Integrated Generic Resources: Product Structure Configuration. This part addresses all major aspects of product structure and configuration management including: definition of part versions; release and approval; assemblies; configuration management of assemblies, etc. There are also other parts that relate at least partially to the topic of product structuring. These include:

- iso 10303-41 Integrated Generic Resources: Fundamentals of Product Description and Support;
- iso 10303-42 Integrated Generic Resources: Geometric and Topological Representation; and
- iso 10303-43 Integrated Generic Resources: Representation Structures (also referred to as the product shape integration model).

It should be noted that implementation of the standard in the form of CAD systems and other

information technology may reflect different divisions or packaging than the parts of the standard itself. For example, it is envisioned that a STEP system for a manufacturing enterprise might have system components for: features analysis/feature-based design, geometry development; mechanical CAD, electrical CAD, finite element modeling, BOM processing, versioning/configuration management, material forecasting, and manufacturing process development.

The parts of the standard noted above are *generic* data models. There are also resource models that are intended to provide information in generalized application domains. They draw on the generic models, specializing them as necessary. Examples are models for finite element analysis and kinematics. Additionally, there are application protocol data models that target specific application domains. Examples are sculptured surface models and two-dimensional drafting models. These can be furthered specialized to application areas such as mechanical, electronics, architectural, and process industries.

Data models comprising the STEP standard can also be categorized based on the level at which they address their subject area. These categories can be described as: data within a part; data about a part; and data about a group of parts. 'Data within a part' is the data that is used to describe the physical nature of the part itself, such as its topology and geometry. 'Data about a part' is related to the part, but is not directly associated with the description of the nature of the part. This data includes, for example, part name and part number, security classification, approvals, revision status, material, etc. Some of this data is data that might typically be associated with a BOM, particularly a BOM for a component part. 'Data about a group of parts' focuses on assemblies. Examples of this type of data include which versions of what parts go into an assembly and where each component part or subassembly is located in the context of the assembly space. In other words, this data addresses traditional parts lists or engineering BOMs, but also contains additional information relating to component usage or location on the assembly. That additional data was noted as being useful to manufacturing personnel for actually assembling the part. In this sense, the STEP concept of an engineering BOM can include additional data intended to assist with the manufacturing of the part.

The STEP standard supports different representations of a part, but these differences seem to relate largely to scale and form, i.e., how much detail is provided or what presentation format is used. For example, it is noted that most design engineers and manufacturing engineers need a detailed product model. However, in cases where the product model is exchanged with a customer or supplier, some details of the part structure may be hidden to protect proprietary part designs. It was also noted that product models and related documentation used for support activities, such as maintenance, only need to show replaceable part units and can often be presented in simple 2-D form rather than as 3-D solid models.

In summary, the STEP standard is intended to produce a single, logical product model that can be consistently accessed and shared across the various activities performed throughout the life cycle and their supporting information systems. STEP does not preclude allowing specialized data to be layered onto core generic data models, and in fact, supports this approach through a deliberately layered data architecture. By the same token, the standard does not provide any explicit support for encoding and relating multiple BOMs. Also, the STEP standard in its general tone seems to encourage an overall concept of part data standardization (in the sense of reducing any unnecessary variability) across the life cycle, coupling that with the concept of multiple presentations ('views'). This seems to indicate a preference for creating a common BOM with some (albeit maybe limited) specialization and with multiple presentations.

STEP AND MULTIPLE BOMS

The STEP product structure model — as represented in logical or abstract form in ISO 10303 Parts 41, 43 (minimally) and 44, and physically (i.e., 'implementationally') in AP 203 (and to some degree in AP 208 as well) — is officially neutral with respect to multiple BOMs. Specifically, the STEP view is some enterprises allow them, some don't, so the STEP model should support representation of multiple BOMs, but within the same basic model framework as the basic BOM (i.e., no or only minimal data structures should be defined to explicitly support BOM multiplicity, and anything that does support this approach should not affect or otherwise change the other portions of the data model).

The original data entity STRUCTURE was replaced by another entity called PRODUCT_DEFINITION,

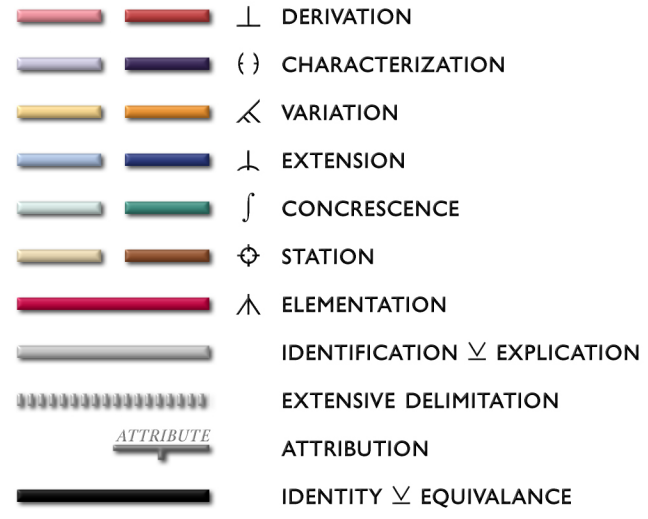
which as one would expect is the top-level product structure entity. It basically does the same thing as `STRUCTURE`, only it also has some other general data elements that were deemed appropriate for the highest-level entity. As with `STRUCTURE`, one can establish pointers, or at a minimum a system implemented based on this standard could determine that there were multiple BOMs based on the fact that there would be many (more than one at least) `PRODUCT_DEFINITION` entities for the same part. (Note that how “same” The model also still contains `MAKE_FROM_USAGE`, which is the main data entity intended to deal both with raw material and ‘factorization’ issues.

These data entities are in Parts 41 and 44 of ISO 10303, but not in AP 203, which is intended to be much narrower in focus (they deliberately did not want to get into the multiple BOM issue in this AP). AP 203 is both a specialization and application of Parts 41 and 44. AP 203 is specialized in the sense that in order to apply Parts 41 and 44, certain things in the abstract data models were left out or further restricted. For example, all the parent-child relations or constraints in AP 203 are binary, i.e., they do not allow for the additional conventions introduced to handle other-than-next-higher assembly pointers or similar conditionals. Something like 60+ additional rules or constraints were introduced into the model to make it ‘implementable’. Rule 15 is the closest rule dealing with the multiple BOM issue—but it really addresses versioning or configuration management issues, not multiple BOMs. AP 203 does not go very far into configuration management, however. The AP that is most directly tackling this issue (and may have some multiple BOM-related data structures based on versioning `PRODUCT_DEFINITION`) is AP 208 Life-Cycle Change Management. As of this writing, AP 208 is just now out for comment in draft form.

ATTACHMENT 2: A2 FORMAT TECHNICAL DRAWINGS

Figure 29. Taxon Phylogeny and Taxonomy

LEGEND



NOTES

1. This drawing is sheet 1 of 8 which collectively depict the principal elements of Ontek metasytematics. This particular drawing depicts the phylogenetic derivation of the fact *Taxis*, together with its metasytematic variants, possible extensional structures, and coordinate categorical constructs. The schematic elements of this drawing depicting axial and extensive elements of these structures (the grey boxes) are incomplete and hence not as yet definitive.

—TO BE COMPLETED AT A LATER DATE—

2. The METATYPE attribute of taxis PHYLE_X^n indicates the *inframetataxic configuration* of the metasystematic configuration PHYLE. Possible variants of these are:

- PHYLE
- SUBPHYLE
- DEME
- SUPERPHYLE
- ALLOPHYLE
- HOMOTAXIC MULTPHYLE
- HETEROTAXIC MULTPHYLE

—TO BE COMPLETED AT A LATER DATE—

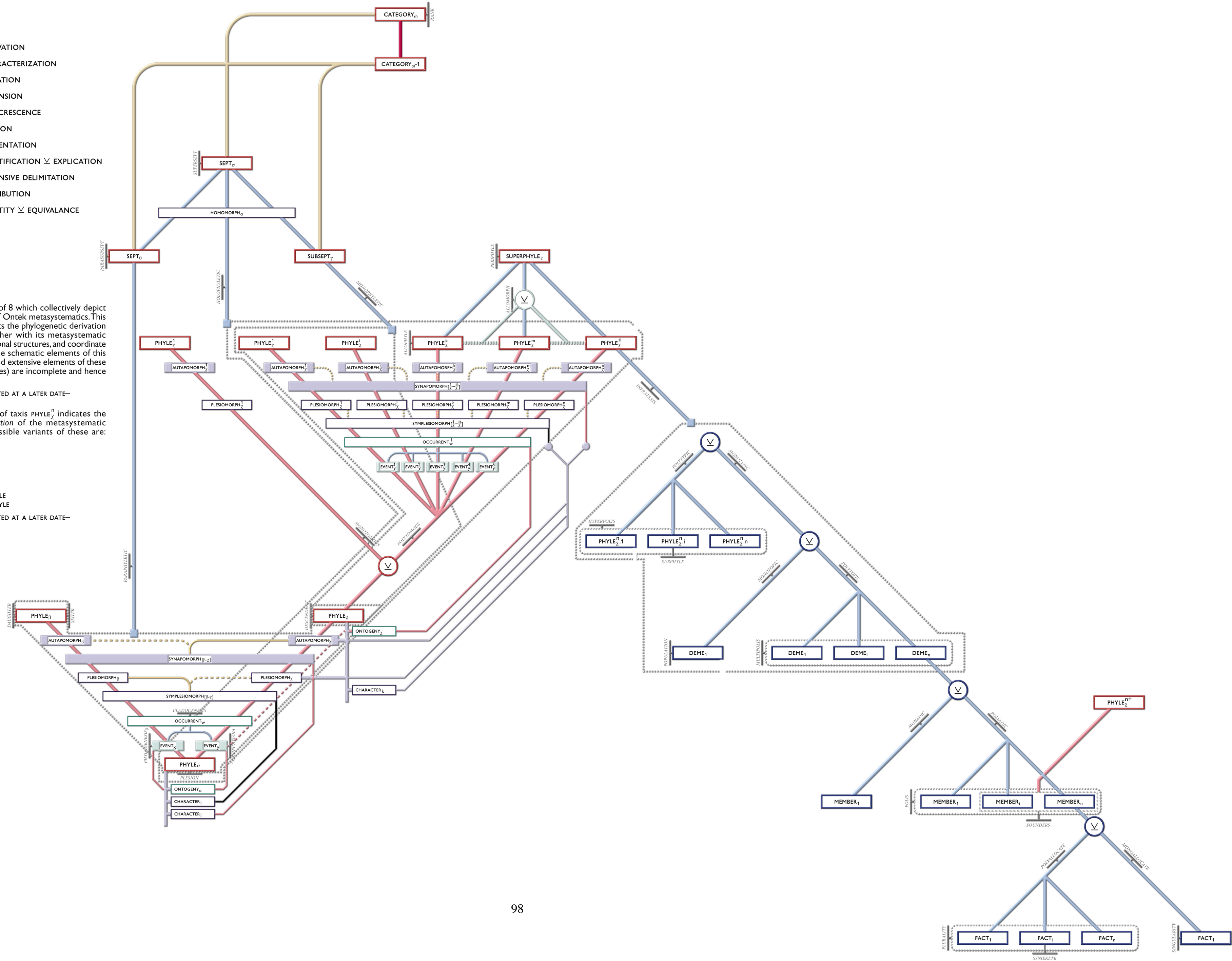


Figure 30. Architecture Element Elements

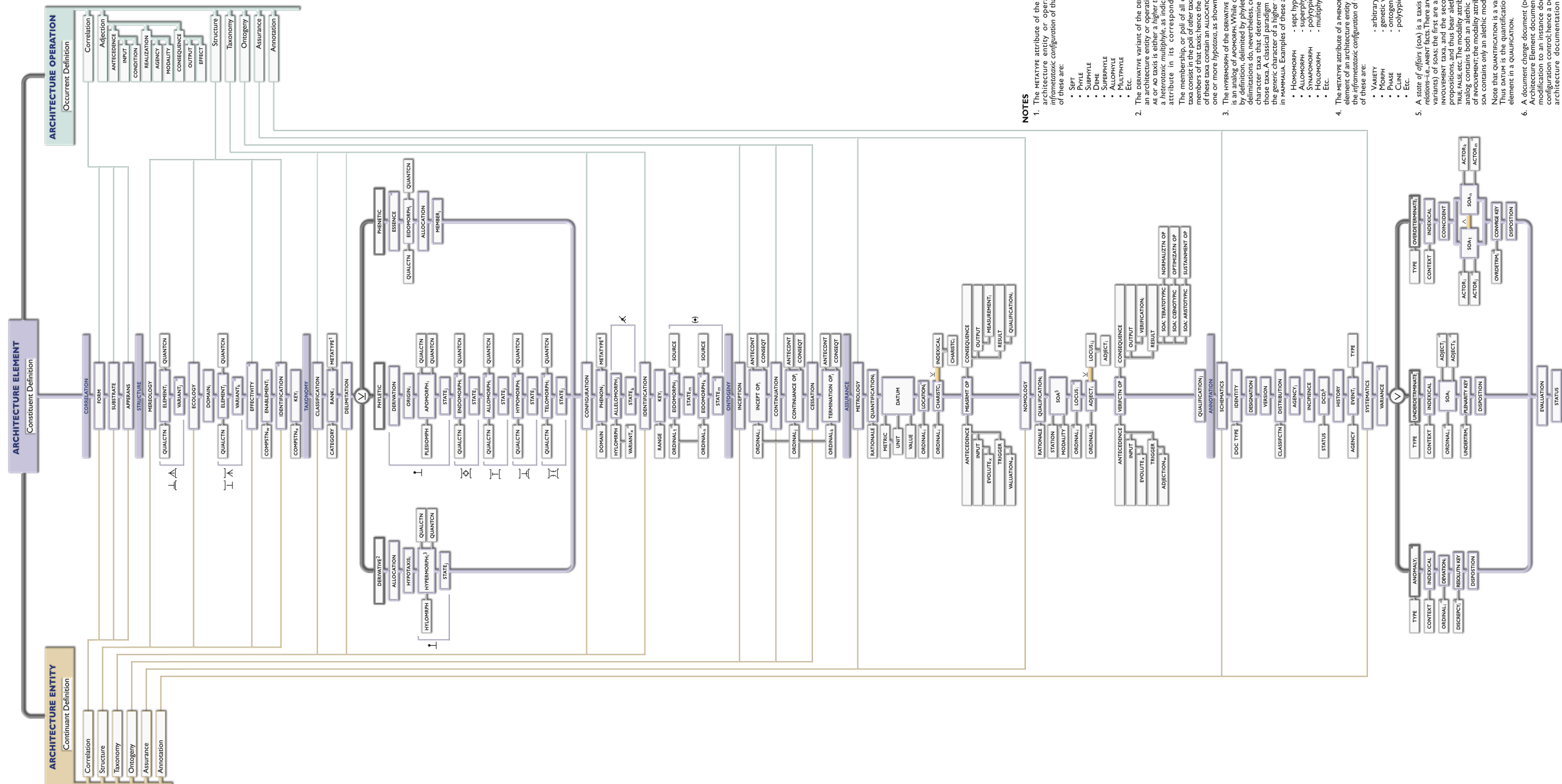


Figure 31. Antecedence and Consequence Taxonomy

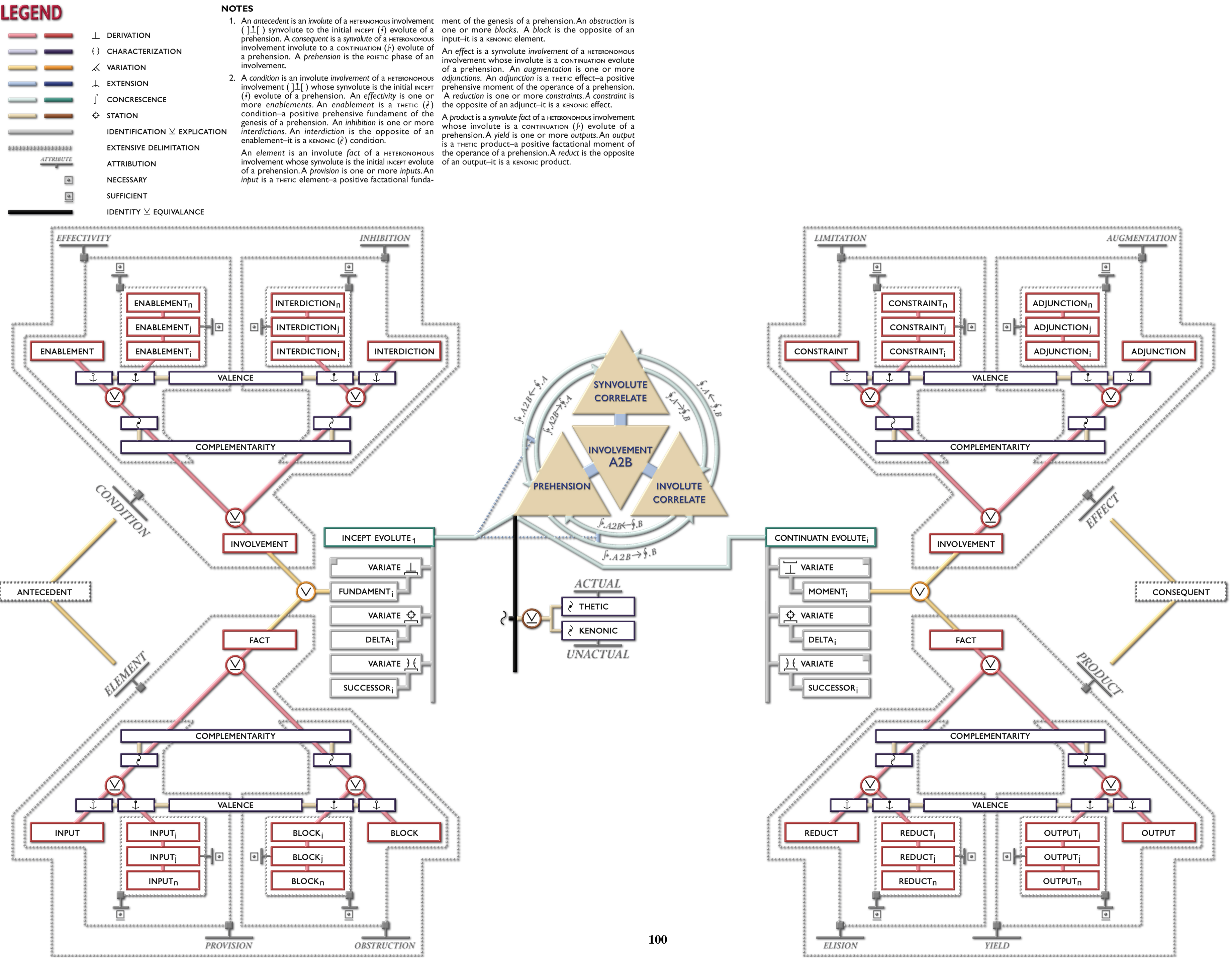


Figure 32. Similarity Phylogeny and Taxonomy

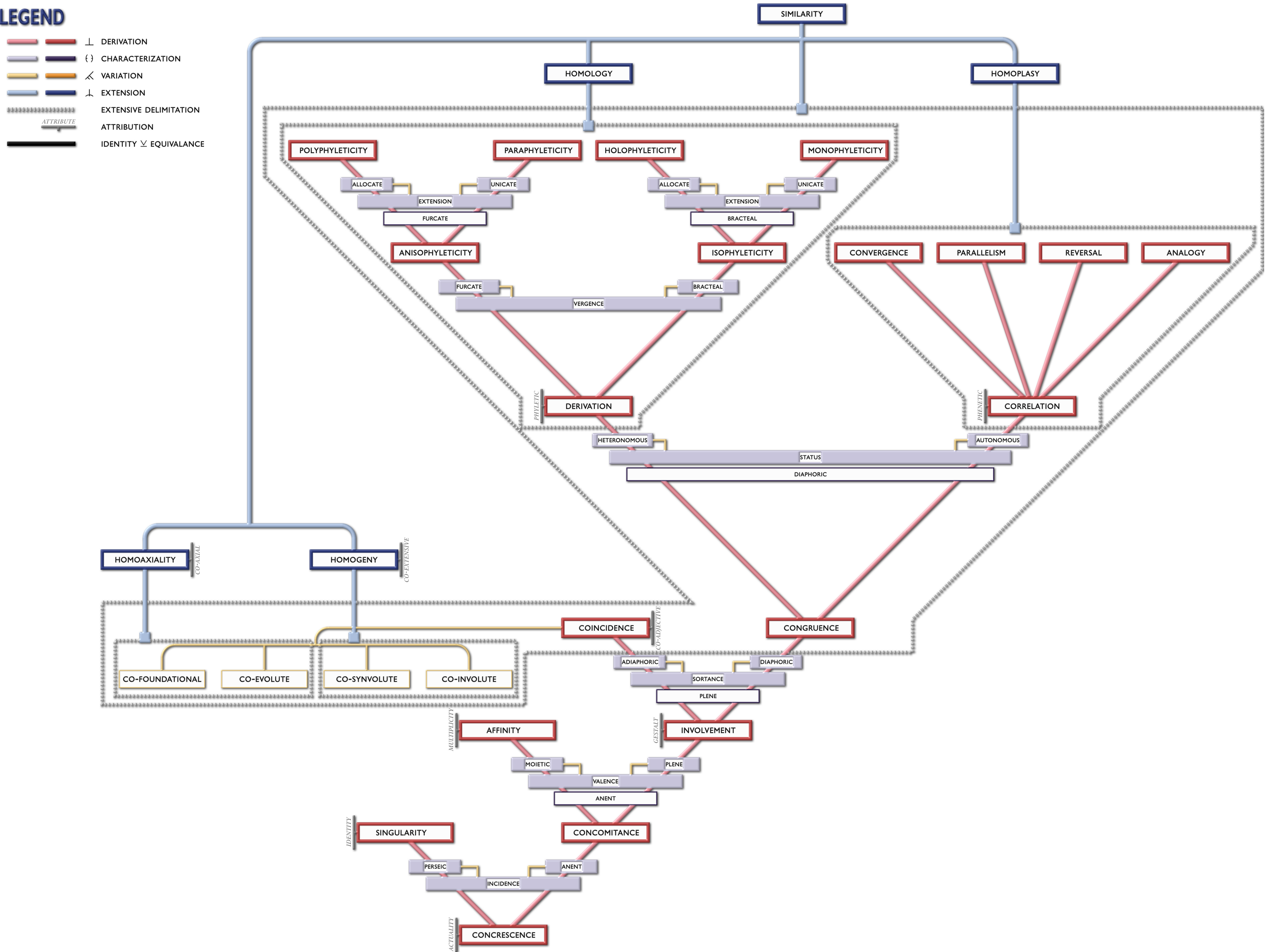


Figure 33. Metastructure Elements and Examples

	TYPE	RELATION OF	RELATION BETWEEN↔	DELIMITS	EXPLICATES...; ENABLES REPRESENTATION OF...	INTRODUCES...; ENTAILS...	PRODUCT STRUCTURE EXAMPLES	ENTERPRISE ENGINEERING EXAMPLES
CORE	1	COMPOSITION	Containment	Entities↔Elements	MEROLOGY	STRUCTURE	SYSTEMS↔SUBSYSTEMS; ASSEMBLIES↔COMPONENTS	ENTERPRISES↔{SUBSIDIARIES,DIVISIONS,ORGANIZATIONS}
	2	CONSTITUTION	Materiality	Entities↔Materials	SUBSTANTIATION	MATERIALIZATION	PARTS↔RAW MATERIALS; COMPOUNDS↔CHEM ELMTS	OPSYSTEMS↔{FISCAL ASSETS,INFORMATION,ORGS}
	3	INHERENCE	Characterization	Entities↔Attributes	MORPHOLOGY	PARTICULARITY; FEATURES, PROPERTIES AND DISPOSITIONS	PARTS↔FEATURES; SYSTEMS↔PERFORMANCE CHARS	OPSYSTEMS↔{CAPABILITY,RELIABILITY,EFFICIENCY}
	4	QUALIFICATION	Conditionality	Relations↔Antecedents	NOMOLOGY	CONDITIONALITY; OPERANT/INOPERANT CONDITIONS	NEEDS↔REQMTS; PROCS↔INPUTS; COMPSNTS↔EFFIES	{STAKEHOLDER PURPOSE ₁ ,... _n }↔BALANCED SOLUTIONS
	5	QUANTIFICATION	Multery	Relata↔Quantities	METROLOGY	MAGNITUDE; NUMERICAL CONDITIONS AND MEASUREMENT	TPMS↔TPM VALUES; WHERE-USED QUANTITIES	FISCAL METRICS↔VALUES; PRODUCTS↔MARKET SHARES
INTRINSIC	6	EQUIVALENCE	Substitutionality	Relata↔Substituends	ISOLOGY	FUNCTIONAL IDENTITY; INSTRUMENTAL EFFICIENCIES	MIL-P-18177 TYPE G10↔LP-509 CLASS II GRADE A	MIL-Q-9858↔ISO9000; SUPPLIER ₁ ↔SUPPLIER _n
	7	ALTERNATION	Optionality	Relata↔Alternatives	HETEROLOGY	FUNCTIONAL CONGRUENCE; APPLICATIVE EFFICIENCIES	F-16 AVIONICS↔COUNTRY-SPECIFIC PACKAGES	MATERIEL↔{MAKE,BUY}; STRAT PGM↔{CONTINGENCY ₁ ,... _n }
	8	VARIATION	Diversity	Baselines↔Variants	CONFIGURATION	MULTIFORMITY; ADAPTATIONAL AMPLITUDE AND APPLICATIVE SCOPE	JSF↔{AF VARIANT,STOL VARIANT}	PRODUCT REALZTN↔{DOD VARIANT,COMRCL VARIANT}
	9	ORDER	Positionality	Entities↔Positions	ONTONOMY	RELATIVE LOCATION; SPATIAL AND TEMPORAL ORDERING	ASSEMBLY ORDER; SERVICE PRECEDENCE	ACTION ₁ ↔ACTION ₂ ; OBJECTIVES↔PRIORITIES
	10	TRANSFORMATION	Development	Entities↔States	ONTOGENY	CHANGE; TEMPORAL VERSIONS AND ANOMALY RESOLUTIONS	C-130H↔C-130J; LINUX KERNEL V2.0.3↔V2.0.4	COLD WAR A&D↔POST-CW A&D; EPA REGS v1999↔2000
ADJUNCT	11	ARTICULATION	Separation	Entities↔Realizations	MODALITY	PARTITIONED REALIZATION; CONSTRAINT CIRCUMVENTIONS	NOZZLE ASSEMBLY↔{THROAT,CHAMBER,EXIT CONE}	ENTERPRISES↔{STRATEGIC BUSINESS UNIT ₁ ,... _n }
	12	FACTORIZATION	Abstraction			MUTIPLXED REALIZATION; REDUNDANCY ELIMINATIONS	{GEAR ₁ ,GEAR _n }↔GEAR BLANK	{TASK ^{typeA} ₁ ,... _n }↔FUNCTIONAL ORG
	13	CONSOLIDATION	Integration			AGGREGATE REALIZATION; MODULARITY ENHANCEMENTS	{CHIPS,BOARDS,HOUSING}↔AVIONICS LRU	{TASK ^{typeX} ₁ ,... _n }↔PROCESS ORG
EXTRINSIC	14	INVOLVEMENT	Participation	Continuants↔Occursents	REALIZATION	DYNAMICS; OBJECT/PROCESS GESTALTS	PRODUCTS↔FUNCTIONS; ATTRIBUTES↔TEST PROCS	OPSYSTEMS↔ENT PROCS; FISCAL ATTRS↔AUDIT PROCS
	15	CAPACITATION	Provisionment	Involvements↔Tools;Agencies	FACILITIZATION	INSTRUMENTALITY; FUNCTIONAL AND AGENTIAL ENABLERS	F-18E/F ASSEMBLY↔{FIXTURES; JIGS; ASSEMBLERS}	OPSYSTEMS↔{FORMAL METHODS,INTEL,HUMAN ASSETS}
	16	REPRESENTATION	Objectification	Entities↔Representations	PHENOMENOLOGY	REPRESENTATION; INTENTIONAL FORM	PRDCTS↔{ENG DWGS,DATA}; PROCS↔{TECH MNLS,CODE}	INTENTS↔MISSION STATEMNTS; RULES↔POLICY STMTS
	17	DESIGNATION	Nominalization	Entities↔Designators	SYMBOLOLOGY	NOMENCLATURE; SYMBOLIC FORM AND NOMINAL REFERENCE	PARTS↔PART NUMBERS; AIRCRAFT↔TAIL NUMBERS	ENTERPRISES↔LOGOS; PRODUCTS↔BRAND NAMES
INTERTYPEIC	18	CORRELATION	Complementation	Taxa↔Complements	ONTOLOGY	METASTRUCTURE; FACTS OF FORM AND FORMS OF FACTS	METASCHMATIC ARTICULATION & SELF-APPLICATION; '3-SCHEMA ARCHITECTURE'	ENTERPRISES↔{ARCHITECTURES,OPSYSTEMS,PRODUCTS}
	19	DELIMITATION	Differentiation	Taxa↔Characters	PHYLOGENY	GROUPS; FACTS OF SIMILARITY AND GENERIC RELATEDNESS	SCHEMATIC REFLECTION & METATYPE MULTIPLICITY	PROCESSES↔MOES; PRODUCTS↔COST/QUAL/PERFMNCE
	20	IDENTIFICATION	Membership	Taxa↔Keys	TYPOLGY	INDIVIDUALS; FACTS OF RESEMBLANCE AND SPECIFIC RELATEDNESS	PHENON/TAXON DISTINCTION, INSTANCE REFLECTION & TYPE MULTIPLICITY	INDUSTRIES↔SIGNATURES; AGENCIES↔BEHAVIORS
	21	CLASSIFICATION	Rank	Taxa↔Categories	STATUS	SCHEMATIC LOCATION; FACTS OF RETICULATION AND INDEXICAL POSITION	CATEGORY/TAXON DISTINCTION & COORDINATE MULTIPLICITY	POPULATIONS↔{SUBSPECIES,SPECIES,GENUS,...,DOMAIN}
	22	DEVIATION	Irregularity	Delimitations↔Anomalies	TERATOLOGY	ABERRATION; EXCEPTION CONDITIONS	INDETERMINANCE & INOPERABILITY; DETECTION & RESOLUTION MECHANISMS	PRODUCTS↔DEFECTS; PROCESSES↔CAPABILITY GAPS
TAXONOMIC								

